Red Hat Reference Architecture Series

Red Hat Cloud Foundations Reference Architecture

Edition One: Automating Private IaaS Clouds on Blades

Version 1.0
September 2010

Red Hat Cloud Foundations Reference Architecture
Edition One: Automating Private IaaS Clouds on Blades

1801 Varsity Drive™
Raleigh NC 27606-2072 USA
Phone: +1 919 754 3700
Phone: 888 733 4281
Fax: +1 919 754 3701
PO Box 13588
Research Triangle Park NC 27709 USA

The information contained herein is subject to change without notice. Red Hat, Inc. shall not be liable for technical or editorial errors or omissions contained herein.

The GPG fingerprint of the security@redhat.com key is:
CA 20 86 86 2B D6 9D FC 65 F6 EC C4 21 91 80 CD DB 42 A6 0E

Table of Contents

1 Executive Summary..6

2 Cloud Computing: Definitions..7

2.1 Essential Characteristics..7

2.1.1 On-demand Self-Service 7

2.1.2 Resource Pooling...7

2.1.3 Rapid Elasticity 7

2.1.4 Measured Service...7

2.2 Service Models...8

2.2.1 Cloud Infrastructure as a Service (IaaS).......................................8

2.2.2 Cloud Platform as a Service (PaaS)..8

2.2.3 Cloud Software as a Service (SaaS)..8

2.2.4 Examples of Cloud Service Models..9

2.3 Deployment Models..10

2.3.1 Private Cloud..10

2.3.2 Public Cloud...11

2.3.3 Hybrid Cloud..12

2.3.4 Community Cloud...12

3 Red Hat and Cloud Computing...13

3.1 Evolution, not Revolution – A Phased Approach to Cloud Computing..........13

3.2 Unlocking the Value of the Cloud...15

3.3 Redefining the Cloud..16

3.3.1 Deltacloud..16

4 Red Hat Cloud: Software Stack and Infrastructure Components...................18

4.1 Red Hat Enterprise Linux...19

4.2 Red Hat Enterprise Virtualization (RHEV) for Servers 20

4.3 Red Hat Network (RHN) Satellite..21

4.3.1 Cobbler...21

4.4 JBoss Enterprise Middleware...22

4.4.1 JBoss Enterprise Application Platform (EAP)................................23

4.4.2 JBoss Operations Network (JON)..23

4.5 Red Hat Enterprise MRG Grid..24

5 Reference Architecture System Configuration...25
5.1 Server Configuration...26
5.2 Software Configuration...27
5.3 Blade and Virtual Connect Configuration...28
5.4 Storage Configuration 28
5.5 Network Configuration..29
6 Deploying Cloud Infrastructure Services...30
6.1 Install Management Build-Up System..31
6.1.1 Common Scripts and Utilities..31
6.1.2 Prepare the Build-Up Installation...33
6.1.3 Install...39
6.2 Create Satellite System...40
6.2.1 Prepare Host..40
6.2.2 Install VM for Satellite..47
6.2.3 Install and Configure RHN Satellite Software..52
6.2.4 Post Satellite Installation Actions...56
6.3 Provision First Management Node...78
6.4 Creating Management Virtual Machines..93
6.4.1 JON VM 93
6.4.2 MRG VM..101
6.4.3 RHEV-M VM...107
6.5 Provision Second Management Node...107
6.6 Create First Hosts...109
6.6.1 RHEL Configuration for a RHEV Host...119
6.7 Configuring RHEV...121
6.7.1 Installing RHEV Manager Software...121
6.7.2 Configure the RHEV Environment...122
6.7.3 Upload ISO Images..125
7 Dynamic Addition and Removal of Hosts...126
7.1 RHEL / KVM Host Addition...126
7.2 RHEV Host Addition..128
7.3 Host Removal..131
8 Creating VMs...134
8.1 RHEL...135

8.2 RHEL with Java Application....136
8.3 RHEL with JBoss....137
8.4 RHEL MRG Grid Execute Nodes....140
8.5 RHEL MRG Grid Rendering....145
9 References....148
Appendix A: Configuration Files, Images, Etc....149
A.1 Management Nodes....149
A.1.1 multipath.conf.template....149
A.1.2 cluster.conf....150
A.1.3 named.conf.slave....151
A.2 Satellite 152
A.2.1 answers.txt....152
A.2.2 AppPGP....155
A.2.3 DNS database files....155
A.2.4 dhcpd.conf....156
A.2.5 named.conf....158
A.2.6 resolv.conf....158
A.2.7 settings....159
A.3 RHEV Image File....166
A.4 MRG....168
A.4.1 condor_config.local.mgr....168
A.4.2 condor_config.local_mrgexec....169
A.4.3 condor_config.local_mrgexecrender....170
A.4.4 cumin.conf....171
A.4.5 pg_hba.conf....171
A.4.6 Perfect....172
A.4.7 Blender....173
A.4.8 render.tgz....173
A.5 JBoss....175
A.5.1 JBoss EAP....175
A.5.2 JBoss Seam....175
A.5.3 JON....175
A.6 Blade and Virtual Connect Configuration....178
A.7 Java....197
Appendix B: Bugzillas....199

1 Executive Summary

Red Hat's suite of open source software provides a rich infrastructure for cloud providers to build public/private cloud offerings.

This reference architecture performs many of the tasks detailed in the previously released **Red Hat Cloud Foundations, Edition One: Private IaaS Clouds**. In this reference architecture, the application programming interfaces (APIs) and command line interfaces (CLIs) are used in the deployment of the infrastructure. Using APIs, CLIs, and scripts enables automation of the installation and configuration of systems and software (e.g. Red Hat Enterprise Linux, Red Hat Network Satellite, cobbler, hypervisors, tenant applications, etc.) opposed to requiring manual actions for each of the cloud foundation elements.

Additionally, the server platform is a blade system, which is ideal for the dense deployments of a cloud infrastructure. The topics addressed in this reference architecture are the same as the predecessors:

1. Deployment of infrastructure management services, e.g., Red Hat Network (RHN) Satellite, Red Hat Enterprise Virtualization (RHEV) Manager (RHEV-M), DNS, DHCP service, PXE server, GFS2 for VM definitions, NFS server for ISO images, JON, MRG Manager - most installed in virtual machines (VMs) in a Red Hat Cluster Suite (RHCS) cluster for high availability.
2. Deployment of a farm of RHEV host systems (either in the form of RHEV hypervisors or as RHEL+KVM) to host VMs deployed by cloud tenants.
3. Demonstrate sample RHEL application(s), JBoss application(s) and MRG Grid application(s) respectively in the tenant VMs.

2 Cloud Computing: Definitions

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of four **essential characteristics,** three **service models**, and four **deployment models**. The following definitions have been proposed by National Institute of Standards and Technology *(*NIST) in the document found at http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc

2.1 Essential Characteristics

Cloud computing creates an illusion of infinite computing resources available on demand, thereby eliminating the need for Cloud Computing users to plan far ahead for provisioning.

2.1.1 *On-demand Self-Service*

A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service's provider.

2.1.2 Resource Pooling

The computing resources of the provider are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify a location at a higher level of abstraction (e.g., country, state, or data center). Examples of resources include storage, processing, memory, network bandwidth, and virtual machines.

2.1.3 *Rapid Elasticity*

Capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time.

2.1.4 Measured Service

Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported providing transparency for both the provider and consumer of the used service.

2.2 Service Models

2.2.1 Cloud Infrastructure as a Service (IaaS)

The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and invoke arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of select networking components (e.g., host firewalls).

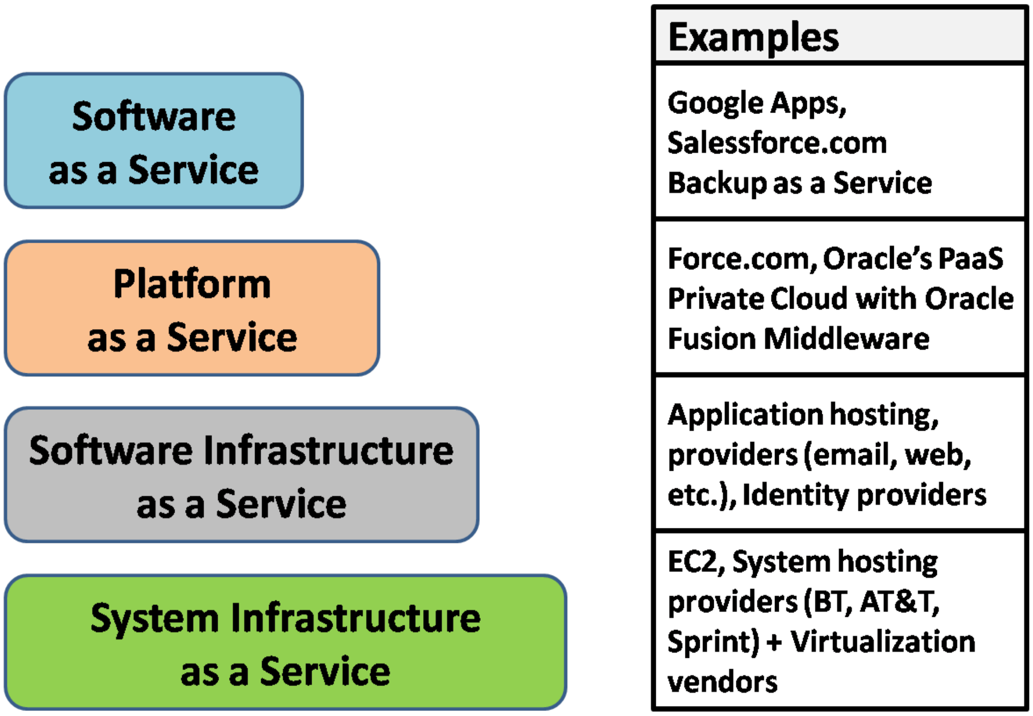
2.2.2 Cloud Platform as a Service (PaaS)

The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly application hosting environment configurations.

2.2.3 Cloud Software as a Service (SaaS)

The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through a thin client interface such as a web browser (e.g., web-based email). The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.

2.2.4 Examples of Cloud Service Models


Software as a Service
Platform as a Service
Software Infrastructure as a Service
System Infrastructure as a Service

Examples
Google Apps, Salessforce.com Backup as a Service
Force.com, Oracle's PaaS Private Cloud with Oracle Fusion Middleware
Application hosting, providers (email, web, etc.), Identity providers
EC2, System hosting providers (BT, AT&T, Sprint) + Virtualization vendors

Figure 1

2.3 Deployment Models

2.3.1 Private Cloud

The cloud infrastructure is operated solely for an organization. It may be managed by the organization or a third party and may exist on or off premise.

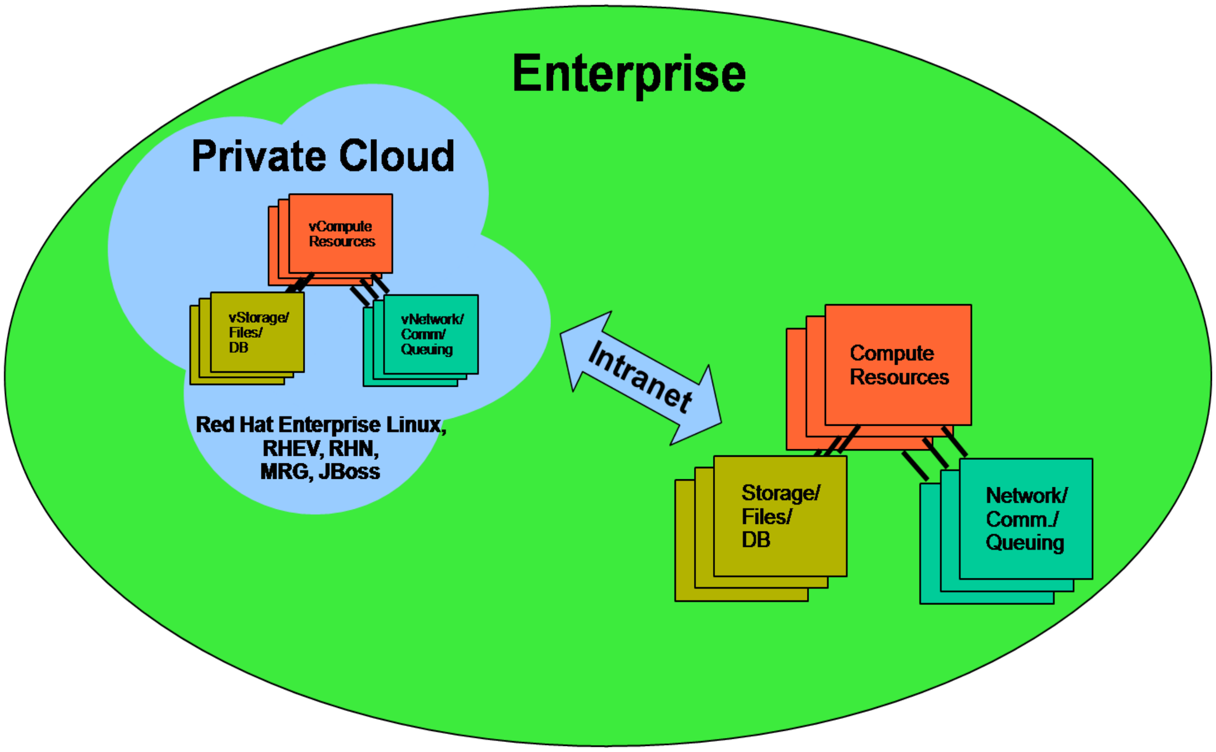


Figure 2

2.3.2 Public Cloud

The cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services.

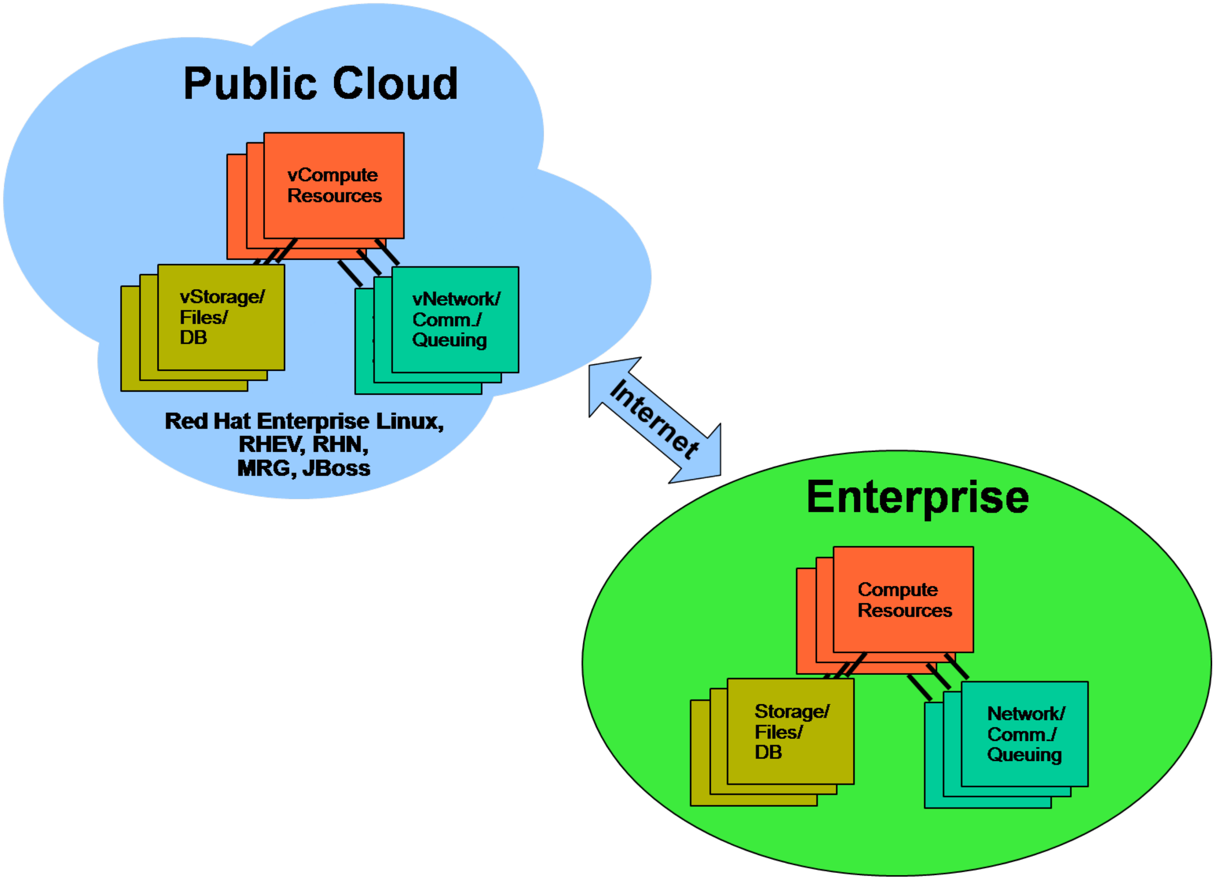


Figure 3

2.3.3 Hybrid Cloud

The cloud infrastructure is a composition of two or more clouds (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability (e.g., load-balancing between clouds).

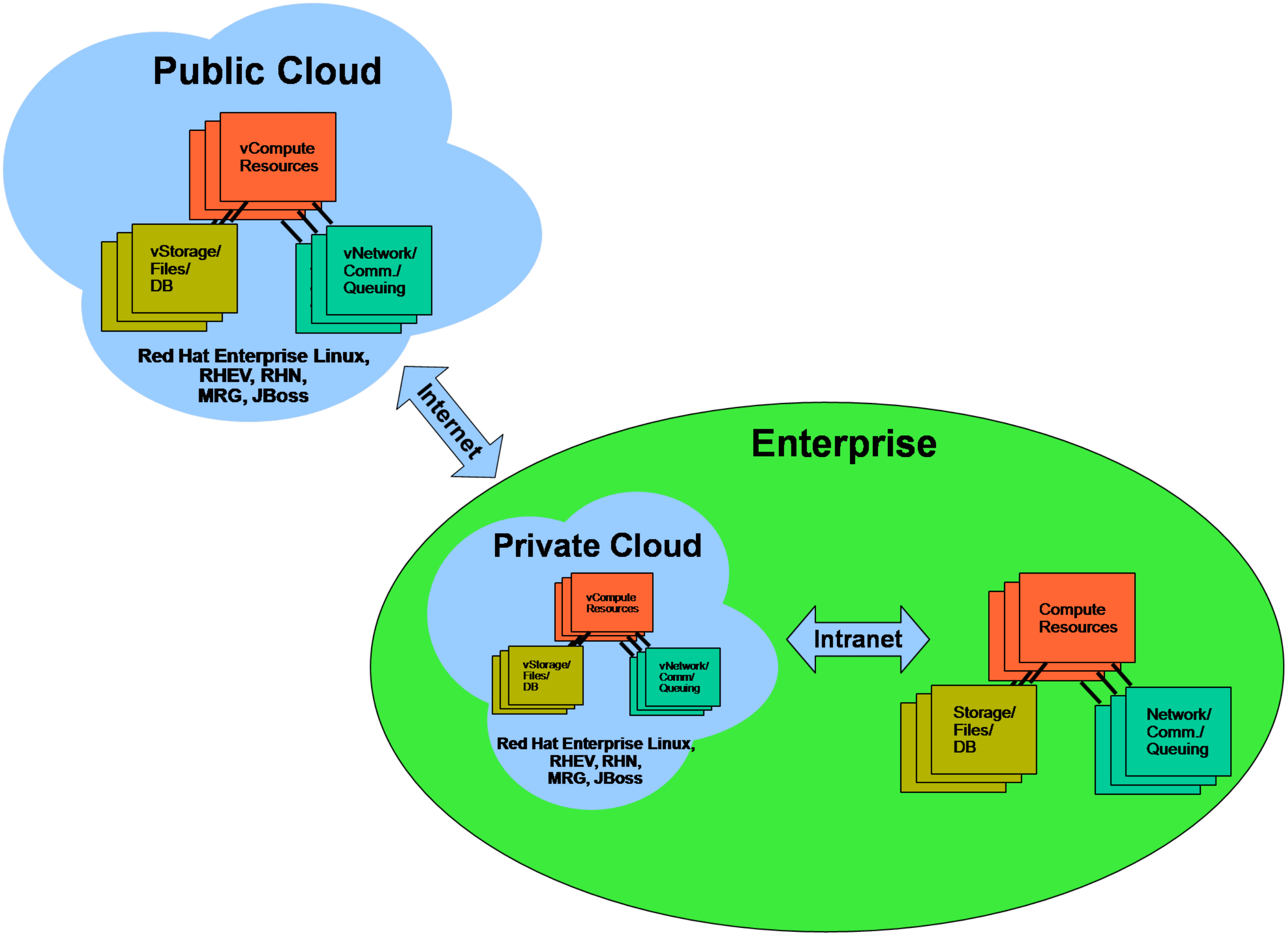


Figure 4

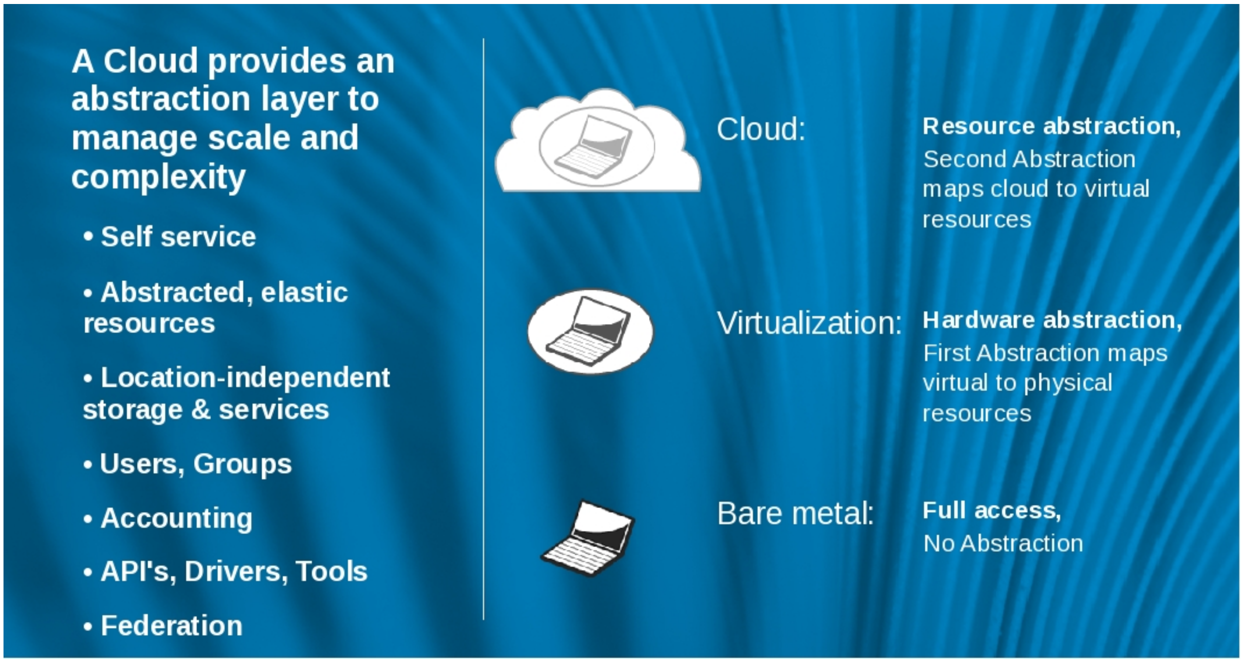
2.3.4 Community Cloud

The cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be managed by the organizations or a third party and may exist on or off premise.

3 Red Hat and Cloud Computing

3.1 Evolution, not Revolution – A Phased Approach to Cloud Computing

While cloud computing requires virtualization as an underlying and essential technology, it is inaccurate to equate cloud computing with virtualization. The figure below displays the different levels of abstraction addressed by virtualization and cloud computing respectively.



A Cloud provides an abstraction layer to manage scale and complexity

- Self service
- Abstracted, elastic resources
- Location-independent storage & services
- Users, Groups
- Accounting
- API's, Drivers, Tools
- Federation

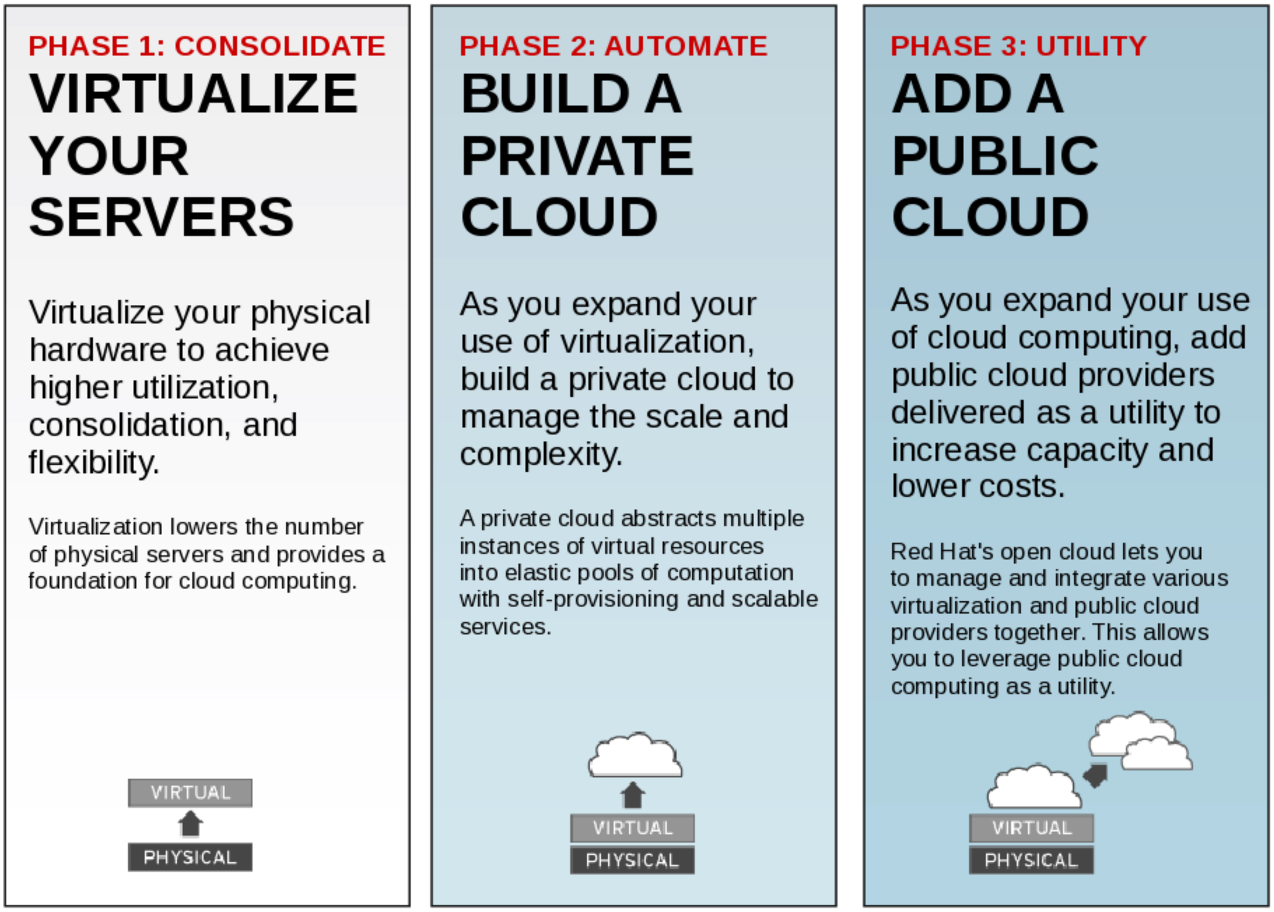
Cloud: **Resource abstraction,** Second Abstraction maps cloud to virtual resources

Virtualization: **Hardware abstraction,** First Abstraction maps virtual to physical resources

Bare metal: **Full access,** No Abstraction

Figure 5: Levels of Abstraction

The following figure illustrates a phased approach to technology adoption starting with server consolidation using virtualization, then automating large deployments of virtualization within an enterprise using private clouds, and finally extending private clouds to hybrid environments leveraging public clouds as a utility.



PHASE 1: CONSOLIDATE

VIRTUALIZE YOUR SERVERS

Virtualize your physical hardware to achieve higher utilization, consolidation, and flexibility.

Virtualization lowers the number of physical servers and provides a foundation for cloud computing.

PHASE 2: AUTOMATE

BUILD A PRIVATE CLOUD

As you expand your use of virtualization, build a private cloud to manage the scale and complexity.

A private cloud abstracts multiple instances of virtual resources into elastic pools of computation with self-provisioning and scalable services.

PHASE 3: UTILITY

ADD A PUBLIC CLOUD

As you expand your use of cloud computing, add public cloud providers delivered as a utility to increase capacity and lower costs.

Red Hat's open cloud lets you to manage and integrate various virtualization and public cloud providers together. This allows you to leverage public cloud computing as a utility.

Figure 6: Phases of Technology Adoption in the Enterprise

3.2 Unlocking the Value of the Cloud

Red Hat's approach does not lock an enterprise into one vendor's cloud stack, but instead offers a rich set of solutions for building a cloud. These can be used alone or in conjunction with components from third-party vendors to create the optimal cloud to meet unique needs.

Cloud computing is one of the most important shifts in information technology to occur in decades. It has the potential to improve the agility of organizations by allowing them to:

1. Enhance their ability to respond to opportunities,
2. Bond more tightly with customers and partners, and
3. Reduce the cost to acquire and use IT in ways never before possible.

Red Hat is proud to be a leader in delivering the infrastructure necessary for reliable, agile, and cost-effective cloud computing. Red Hat's cloud vision is unlike that of any other IT vendor. Red Hat recognizes that IT infrastructure is composed of pieces from many different hardware and software vendors. Red Hat enables the use and management of these diverse assets as one cloud. Enabling cloud to be an evolution, not a revolution.

Red Hat's vision spans the entire range of cloud models:

- Building an internal Infrastructure as a Service (IaaS) cloud, or seamlessly using a third-party's cloud
- Creating new Linux, LAMP, or Java applications online, as a Platform as a Service (PaaS)
- Providing the easiest path to migrating applications to attractive Software as a Service (SaaS) models

Red Hat's open source approach to cloud computing protects and manages existing and diverse investments as one cloud -- whether Linux or Windows, Red Hat Enterprise Virtualization, VMware or Microsoft Hyper-V, Amazon EC2 or another vendor's IaaS, .Net or Java, JBoss or WebSphere, x86 or mainframe.

3.3 Redefining the Cloud

Cloud computing is the first major market wave where open source technologies are built in from the beginning, powering the vast majority of early clouds.

Open source products that make up Red Hat's cloud infrastructure include:

- Red Hat Enterprise Virtualization
- Red Hat Enterprise Linux
- Red Hat Network Satellite
- Red Hat Enterprise MRG Grid
- JBoss Enterprise Middleware

In addition, Red Hat is leading work on and investing in several open source projects related to cloud computing. As these projects mature, after they undergo rigorous testing, tuning, and hardening, the ideas from many of these projects may be incorporated into future versions of the Red Hat cloud infrastructure. These projects include:

- Deltacloud - Abstracts the differences between clouds
- BoxGrinder - A set of projects to build appliances for a multitude of virtualization fabrics
- Cobbler - Installation server for rapid set up of network installation environment
- Condor - Batch system managing millions of machines worldwide
- CoolingTower - Simple application-centric tool for deploying applications in the cloud
- Hail - Umbrella cloud computing project for cloud services
- Infinispan - Extremely scalable, highly available data grid platform
- Libvirt - Common, generic, and scalable layer to securely manage domains on a node
- Spice - Open remote computing solutions for interaction with virtualized desktop devices
- Thincrust - Tools to build appliances for the cloud

3.3.1 Deltacloud

The goal of Deltacloud is simple: making many clouds function as one. Deltacloud strives to bridge the differences between diverse silos of infrastructure, allowing them to be managed as one. Organizations today may have different clouds built on, for example, RHEV-M or VMware vCloud. The Deltacloud project is designed to make them manageable as one cloud, one pool of resources. Organizations may wish to use internal cloud capacity, as well as public clouds like Amazon's EC2, and perhaps capacity from other IaaS providers.

Today each IaaS cloud presents a unique API to which developers and ISVs need to write in order to consume the cloud service. The Deltacloud effort is creating a common, REST-based API, such that developers can write once and manage anywhere. Deltacloud is a cloud broker, so to speak, with drivers that map the API to both public clouds like EC2 and private virtualized clouds based on VMware vCloud, or RHEV-M as depicted in Figure **7**.

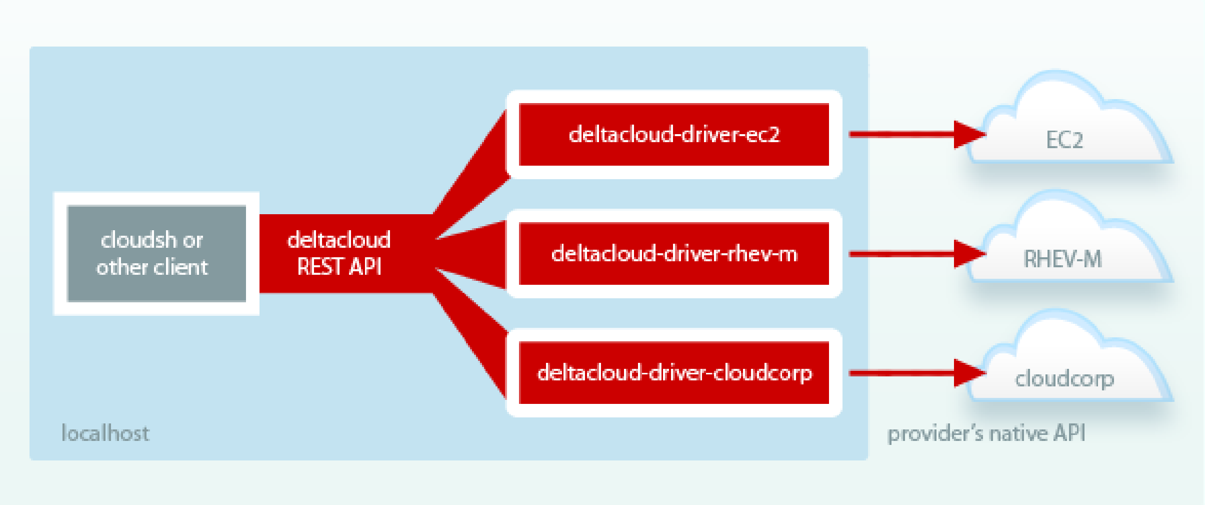


Figure 7: Deltacloud Overview

One level up, Deltacloud Aggregator provides a web UI in front of the Deltacloud API. With Deltacloud Aggregator users can:

- View image status and statistics across clouds, all in one place
- Migrate instances from one cloud to another
- Manage images locally and provision them on any cloud

To learn more about the Deltacloud project, visit http://deltacloud.org.

4 Red Hat Cloud: Software Stack and Infrastructure Components

Figure **8** depicts the software stack of Red Hat Cloud Foundation components.

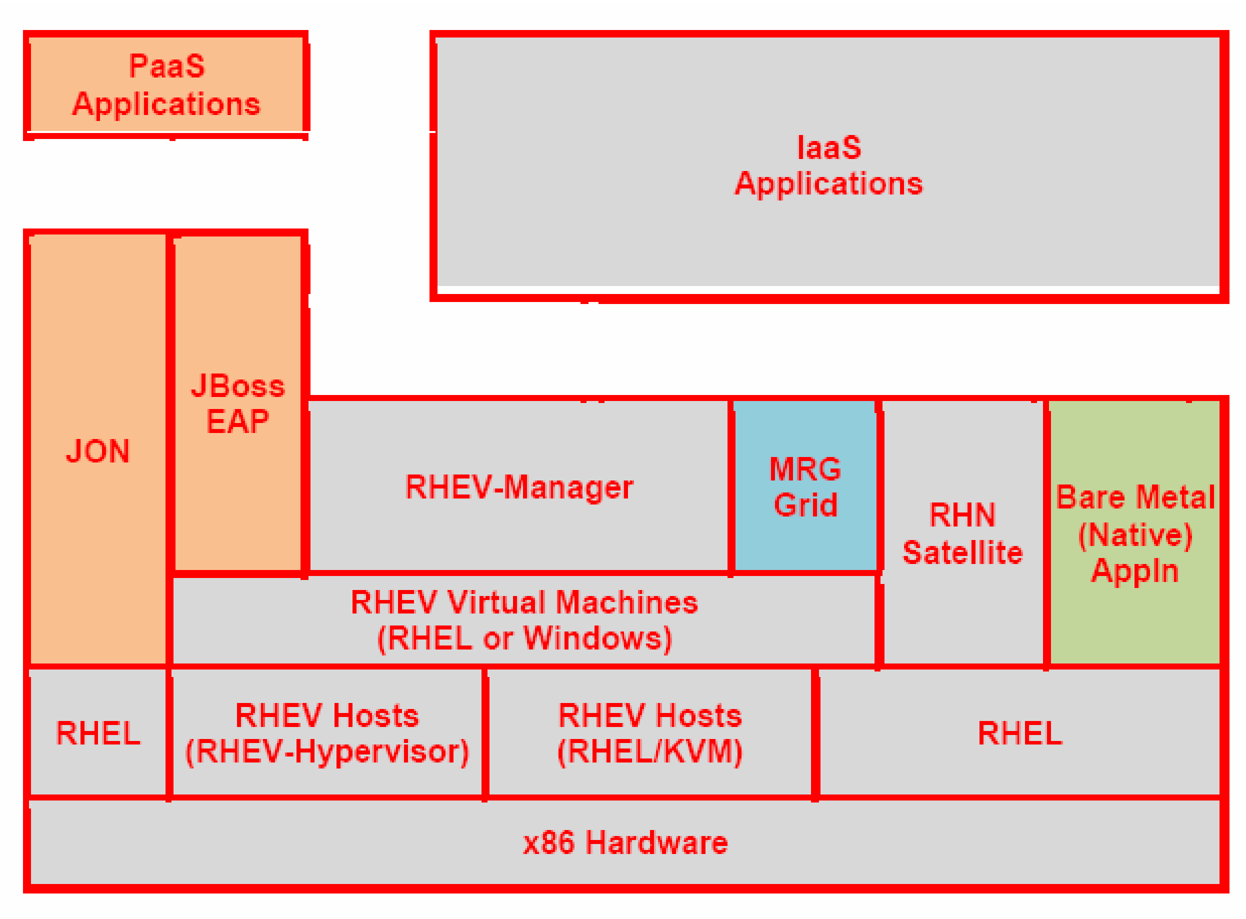


Figure 8: Red Hat Software Stack

4.1 Red Hat Enterprise Linux

Red Hat Enterprise Linux (RHEL) is the world's leading open source application platform. On one certified platform, RHEL offers a choice of:

- Applications - Thousands of certified ISV applications
- Deployment - Including standalone or virtual servers, cloud computing, and software appliances
- Hardware - Wide range of platforms from the world's leading hardware vendors

Red Hat released the fifth update to RHEL 5: Red Hat Enterprise Linux 5.5.

RHEL 5.5 is designed to support newer Intel Xeon® Nehalem-EX platform as well as the AMD Opteron™ 6000 Series platform (formerly code named “Magny-Cours”). The new platforms leverage Red Hat’s history in scalable performance with new levels of core counts, memory and I/O, offering users a very dense and scalable platform balanced for performance across many workload types. To increase the reliability of these systems, Red Hat supports Intel’s expanded machine check architecture, CPU fail-over and memory sparing.

Red Hat also continues to make enhancements to our virtualization platform. New to RHEL 5.5 is support for greater guest density, meaning that more virtual machines can be supported on each physical server. Our internal testing to date has shown that this release can support significantly more virtual guests than other virtualization products. The new hardware and protocols included in the latest release significantly improve network scaling by providing direct access from a guest to the network.

RHEL 5.5 also introduces improved interoperability with Microsoft Windows 7 with an update to Samba. This extends the Active Directory integration to better map users and groups on Red Hat Enterprise Linux systems and simplifies managing file systems across platforms.

An important feature of any RHEL update is that kernel and user application programming interfaces (APIs) remain unchanged, ensuring RHEL 5 applications do not need to be rebuilt or re-certified. The unchanged kernel and user APIs also extend to virtualized environments. With a fully integrated hypervisor, the application binary interface (ABI) consistency offered by RHEL means that applications certified to run on RHEL on physical machines are also certified when run on virtual machines. With this, the portfolio of thousands of certified applications for Red Hat Enterprise Linux applies to both environments.

4.2 Red Hat Enterprise Virtualization (RHEV) for Servers

Red Hat Enterprise Virtualization (RHEV) for Servers is an end-to-end virtualization solution that is designed to enable pervasive data center virtualization, and unlock unprecedented capital and operational efficiency.

RHEV is the ideal platform on which to build an internal or private cloud of Red Hat Enterprise Linux or Windows virtual machines.

RHEV consists of the following two components:

- **Red Hat Enterprise Virtualization Manager (RHEV-M)**: A feature-rich server virtualization management system that provides advanced capabilities for hosts and guests, including high availability, live migration, storage management, system scheduler, and more.
- **Red Hat Enterprise Virtualization Hypervisor**: A modern hypervisor based on KVM which can be deployed as either RHEV-H, a standalone bare metal hypervisor (included with Red Hat Enterprise Virtualization for Servers), or as Red Hat Enterprise Linux 5.4 and later (purchased separately) installed as a hypervisor.

Some key characteristics of RHEV are listed below:

Scalability:

- Host: Up to 512 cores, 1 TB RAM
- Guest/VM: Up to 16 vCPUs, 256 GB RAM

Advanced features:

- Memory page sharing, advanced scheduling capabilities, and more, inherited from the Red Hat Enterprise Linux kernel

Guest operating system support:

- Paravirtualized network and block drivers for highest performance
- Red Hat Enterprise Linux Guests (32-bit & 64-bit): Red Hat Enterprise Linux 3, 4 and 5
- Microsoft® Windows® Guests (32-bit & 64-bit): Windows 2003 server, Windows 2008 server, Windows XP, SVVP, and WHQL certified.

Hardware support:

- All 64-bit x86 servers that support Intel VT or AMD-V technology and are certified for Red Hat Enterprise Linux 5 are certified for Red Hat Enterprise Virtualization.
- Red Hat Enterprise Virtualization supports NAS/NFS, Fibre Channel, and iSCSI storage topologies.

4.3 Red Hat Network (RHN) Satellite

All RHN functionality is on the network, allowing much greater functionality and customization. The Satellite server connects with Red Hat over the public Internet to download new content and updates. This model also allows customers to take their Red Hat Network solution completely off-line if desired. Features include:

- An embedded database to store packages, profiles, and system information.
- Instantly update systems for security fixes or to provide packages or applications needed immediately.
- API layer allows the creation of scripts to automate functions or integrate with existing management applications.
- Distribute custom or 3rd party applications and updates.
- Create staged environments (development, test, production) to select, manage and test content in a structured manner.
- Create errata for custom content, or modify existing errata to provide specific information to different groups.
- Access to advanced features in the Provisioning Module, such as bare metal PXE boot provisioning and integrated network install trees.
- Access to Red Hat Network Monitoring Module for tracking system and application performance.

RHN Satellite is Red Hat's on-premise systems management solution that provides software updates, configuration management, provisioning and monitoring across both physical and virtual Red Hat Enterprise Linux servers. It offers customers opportunities to gain enhanced performance, centralized control and higher scalability for their systems, while deployed on a management server located inside the customer's data center and firewall.

In September 2009, Red Hat released RHN Satellite 5.3, the first fully open source version of the product. This latest version offers opportunities for increased flexibility and faster provisioning setups for customers with the incorporation of open source Cobbler technology in its provisioning architecture.

4.3.1 Cobbler

Cobbler is a Linux installation server that allows for rapid setup of network installation environments. It binds and automates many associated Linux tasks, eliminating the need for many various commands and applications when rolling out new systems and, in some cases, changing existing ones. With a simple series of commands, network installs can be configured for PXE, re-installations, media-based net-installs, and virtualized installs (supporting Xen and KVM).

Cobbler can also optionally help with managing DHCP, DNS, and yum package mirroring infrastructure. In this regard, it is a more generalized automation application, rather than just dealing specifically with installations. There is also a lightweight built-in configuration management system as well as support for integrating with other configuration management systems. Cobbler has a command line interface as well as a web interface and several API access options.

4.4 JBoss Enterprise Middleware

The following JBoss Enterprise Middleware Development Tools, Deployment Platforms and Management Environment are available via subscriptions that deliver not only industry leading SLA-based production and development support, but also includes patches, updates, multi-year maintenance policies, and software assurance from Red Hat.

Development Tools:

- JBoss Developer Studio - PE (Portfolio Edition): Everything needed to develop, test and deploy rich web applications, enterprise applications and SOA services.

Enterprise Platforms:

- JBoss Enterprise Application Platform: Everything needed to deploy, and host enterprise Java applications and services.
- JBoss Enterprise Web Platform: A standards-based solution for light and rich Java web applications.
- JBoss Enterprise Web Server: a single enterprise open source solution for large scale websites and lightweight web applications.
- JBoss Enterprise Portal Platform: Platform for building and deploying portals for personalized user interaction with enterprise applications and automated business processes.
- JBoss Enterprise SOA Platform: A flexible, standards-based platform to integrate applications, SOA services, and business events as well as to automate business processes.
- JBoss Enterprise BRMS: An open source business rules management system that enables easy business policy and rules development, access, and change management.
- JBoss Enterprise Data Services Platform: Bridge the gap between diverse existing enterprise data sources and the new forms of data required by new projects, applications, and architectures.

Enterprise Frameworks:

- JBoss Hibernate Framework: Industry-leading object/relational mapping and persistence.
- JBoss Seam Framework: Powerful application framework for building next generation Web 2.0 applications.
- JBoss Web Framework Kit: A combination of popular open source web frameworks for building light and rich Java applications.
- JBoss jBPM Framework: Business process automation and workflow engine.

Management:

- JBoss Operations Network (JON): An advanced management platform for inventorying, administering, monitoring, and updating JBoss Enterprise Platform deployments.

4.4.1 JBoss Enterprise Application Platform (EAP)

JBoss Enterprise Application Platform is the market leading platform for innovative and scalable Java applications. Integrated, simplified, and delivered by the leader in enterprise open source software, it includes leading open source technologies for building, deploying, and hosting enterprise Java applications and services.

JBoss Enterprise Application Platform balances innovation with enterprise class stability by integrating the most popular clustered Java EE application server with next generation application frameworks. Built on open standards, JBoss Enterprise Application Platform integrates JBoss Application Server, with JBoss Hibernate, JBoss Seam, and other leading open source Java technologies from JBoss.org into a complete, simple enterprise solution for Java applications.

Features and Benefits:

- Complete Eclipse-based Integrated Development Environment (JBoss Developer Studio)
- Built for Standards and Interoperability: JBoss EAP supports a wide range of Java EE and Web Services standards.
- Enterprise Java Beans and Java Persistence
- JBoss EAP bundles and integrates Hibernate, the de facto leader in Object/Relational mapping and persistence.
- Built-in Java naming and directory interface (JNDI) support
- Built-in JTA for two-phase commit transaction support
- JBoss Seam Framework and Web Application Services
- Caching, Clustering, and High Availability
- Security Services
- Web Services and Interoperability
- Integration and Messaging Services
- Embeddable, Service-Oriented Architecture microkernel
- Consistent Manageability

4.4.2 JBoss Operations Network (JON)

JON is an integrated management platform that simplifies the development, testing, deployment and monitoring of JBoss Enterprise Middleware. From the JON console one can:

- inventory resources from the operating system to applications.
- control and audit application configurations to standardize deployments.

- manage, monitor and tune applications for improved visibility, performance and availability.

One central console provides an integrated view and control of JBoss middleware infrastructure.

The JON management platform (server-agent) delivers centralized systems management for the JBoss middleware product suite. With it one can coordinate the many stages of application life cycle and expose a cohesive view of middleware components through complex environments, improve operational efficiency and reliability through thorough visibility into production availability and performance, and effectively manage configuration and rollout of new applications across complex environments with a single, integrated tool.

- Auto-discover application resources: operating systems, applications and services
- From one console, store, edit and set application configurations
- Start, stop, or schedule an action on an application resource
- Remotely deploy applications
- Monitor and collect metric data for a particular platform, server or service
- Alert support personnel based upon application alert conditions
- Assign roles for users to enable fine-grained access control to JON services

4.5 Red Hat Enterprise MRG Grid

MRG Grid provides high throughput and high performance computing. Additionally, it enables enterprises to move to a utility model of computing to help enterprises achieve both higher peak computing capacity and higher IT usage by leveraging their existing infrastructure to build high performance grids.

Based on the Condor project, MRG Grid provides the most advanced and scalable platform for high throughput and high performance computing with capabilities such as:

- scalability to run the largest grids in the world.
- advanced features for handling priorities, workflows, concurrency limits, usage, low latency scheduling, and more.
- support for a wide variety of tasks, ranging from sub-second calculations to long-running, highly parallel (MPI) jobs.
- the ability to schedule to all available computing resources, including local grids, remote grids, virtual machines, idle desktop workstations, and dynamically provisioned cloud infrastructure.

MRG Grid also enables enterprises to move to a utility model of computing, where they can:

- schedule a variety of applications across a heterogeneous pool of available resources.
- automatically handle seasonal workloads with high efficiency, usage, and flexibility.
- dynamically allocate, provision, or acquire additional computing resources for additional applications and loads.
- execute across a diverse set of environments, ranging from virtual machines to bare metal hardware to cloud-based infrastructure.

5 Reference Architecture System Configuration

This reference architecture in deploying the Red Hat infrastructure for a private cloud used the configuration shown in Figure **9** comprised of:

1. Infrastructure management services, e.g., Red Hat Network (RHN) Satellite, Red Hat Enterprise Virtualization Manager (RHEV-M), DNS, DHCP service, PXE server, NFS server for ISO images, JON, MRG Manager - most of which were installed in VMs within a RHCS cluster for high availability.
2. A farm of RHEV host systems (either in the form of RHEV Hypervisors or as RHEL+KVM) to host tenants' VMs.
3. Sample RHEL application(s), JBoss application(s) and MRG Grid application(s) deployed in the tenant VMs.

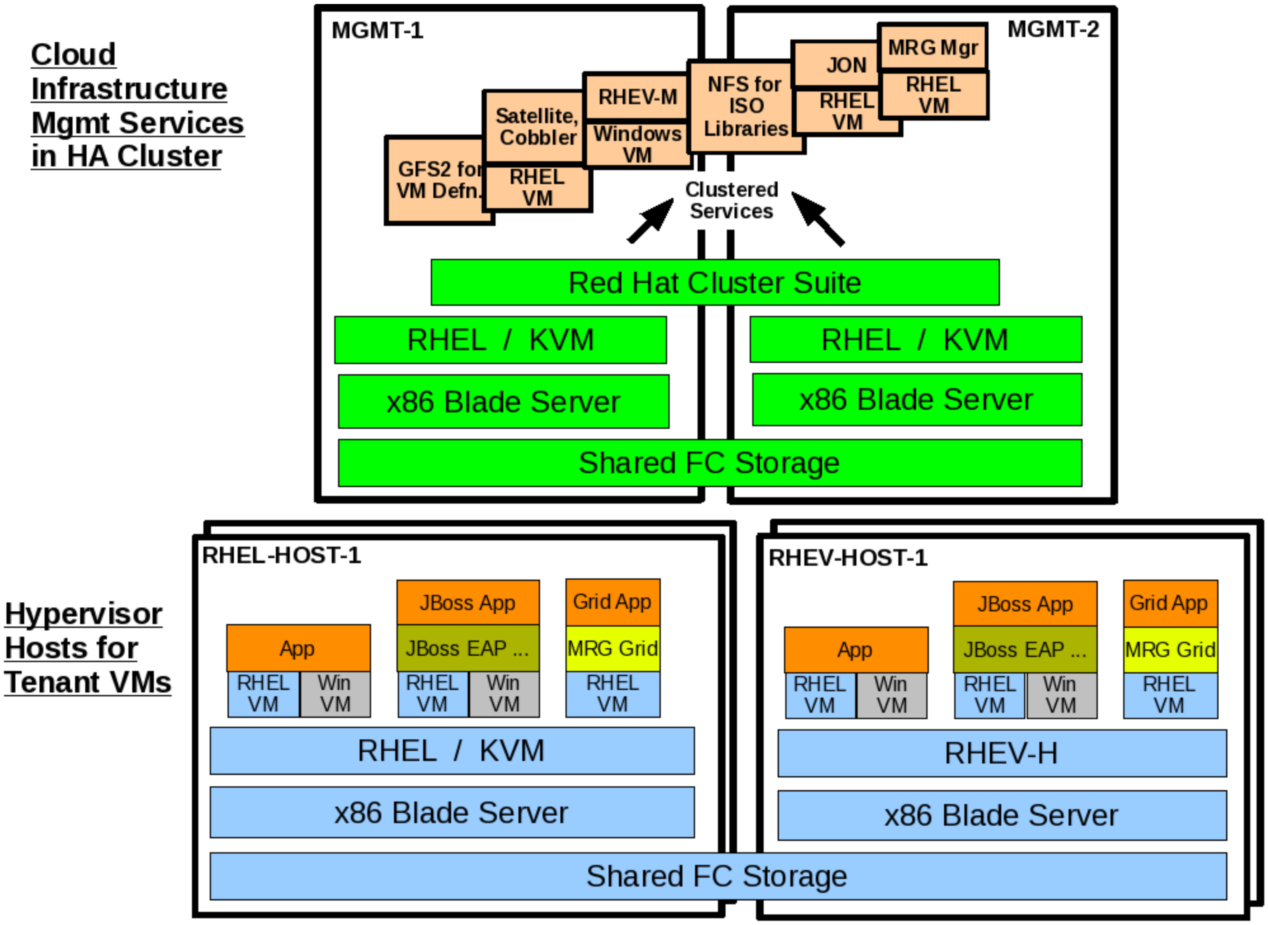


Figure 9

5.1 Server Configuration

Hardware Systems	Specifications
Management Cluster Nodes [2 x HP ProLiant BL460c G6]	Quad Socket, Quad Core (16 cores) Intel® Xeon® CPU X5550 @2.67GHz, 48GB RAM
	2 x 146 GB SATA SSD internal disk drive (mirrored)
	2 x QLogic ISP2532-based 8Gb FC HBA
	2 x Broadcom NetXtreme II BCM57711E Flex-10 10Gb Ethernet Controller
Hypervisor Host Systems [2$^+$ x HP ProLiant BL460c G6]	Quad Socket, Quad Core, (16 cores) Intel® Xeon® CPU W5550 @2.67GHz, 48GB RAM
	2 x 146 GB SATA SSD internal disk drive (mirrored)
	2 x QLogic ISP2532-based 8Gb FC HBA
	2 x Broadcom NetXtreme II BCM57711E Flex-10 10Gb Ethernet Controller

Table 1: Hardware Configuration

5.2 Software Configuration

Software	Version
Red Hat Enterprise Linux (RHEL)	5.5 (2.6.18-194.11.3.el5 kernel)
Red Hat Enterprise Virtualization Manager (RHEV-M)	2.2.0.47069
Red Hat Enterprise Virtualization Hypervisor (RHEV-H)	5.5-2.2 – 6.1
Red Hat Enterprise Linux / KVM (RHEL / KVM)	5.5.0.2 / 83-164
Red Hat Network (RHN) Satellite	5.3.0
JBoss Enterprise Application Platform (EAP)	5.0
JBoss Operations Network (JON)	1.4
Red Hat Enterprise MRG Grid	1.2

Table 2: Software Configuration

5.3 Blade and Virtual Connect Configuration

All the blades are using logical Serial Numbers, MAC addresses and FC WWNs. A single 10Gb network and two 8 Gb FC connections are presented to each host.

Appendix **A.6** has complete details of the blade and virtual connection configuration.

5.4 Storage Configuration

Hardware	Specifications
1 x HP StorageWorks MSA2324fc Fibre Channel Storage Array + HP StorageWorks 70 Modular Smart Array with Dual Domain IO Module [total 49 x 146GB 10K RPM SAS disks]	Storage Controller: Code Version: M110R28 Loader Code Version: 19.009
	Memory Controller: Code Version: F300R22
	Management Controller Code Version: W441R13 Loader Code Version: 12.015
	Expander Controller: Code Version: 1106
	CPLD Code Version: 8
	Hardware Version: 56
1 x HP StorageWorks 4/16 SAN Switch	Firmware: v6.2.2c
1 x HP StorageWorks 8/40 SAN Switch	Firmware: v6.4.0a

Table 3: Storage Hardware

The MSA2324fc array was configured with four 11-disk RAID6 vdisks, with online replacement spares.

LUNs were created and presented as outlined in the following table.

Volume	Size	Presentation	Purpose
MgmtServices	1 TB	Management Cluster	Volume Group for Logical Volumes: • SatVMvol (300GB) • JonVMvol (40GB) • MRGVMvol (40GB) • RHEVMVMvol (30GB) • RHEVNFSvol (300GB)
GFS2	50 GB	Management Cluster	VM Configuration File Shared Storage
RHEVStorage1	1 TB	Hypervisor Hosts	RHEV-M Storage Pool

Table 4: LUN Configuration

5.5 Network Configuration

All the systems in the test environment were assigned to an unique VLAN. Its use allows local control of the network while supplying gateway access to the corporate network. DHCP, DNS, and PXE were all controlled within the VLAN. The VLAN was assigned 10.16.136/21 providing approximately 2000 addresses.

6 Deploying Cloud Infrastructure Services

This section provides the detailed actions performed to configure Red Hat products that constitute the infrastructure used for a private cloud.

The goal is to create a set of highly available cloud infrastructure management services and the initial cloud hosts. The cloud management services and initial hosts are used in the next section to configure additional cloud hosts, VMs within the hosts, and load applications within those VMs.

High availability is achieved by clustering two RHEL nodes using RHCS. Each of the cluster nodes is configured to run RHEL 5.5 with the bundled KVM hypervisor. Each cluster member is provisioned using the environment's RHN Satellite VM. One of the blades (that later becomes a member of the cluster) is temporarily installed with RHEL to host the RHN Satellite with which to provision the cluster members. For the majority of the management services, a VM is created using the KVM hypervisor and configured as an RHCS service. The management service is then installed in the VM (e.g., RHN Satellite VM, JON VM). A high level walk-through of the steps to create these highly available cloud infrastructure management services is presented below.

1. Install RHEL + KVM on a node
2. Create a VM
3. Install RHN Satellite in the VM (Satellite VM)
4. Synchronize Satellite with RHN & download packages from all appropriate channels / child channels:
    - Base RHEL 5
    - Clustering (RHCS)
    - Cluster storage (GFS2)
    - Virtualization (KVM)
    - RHN Tools
    - MRG
    - RHEV management agents for RHEL hosts
5. Use multi-organization support in Satellite - create *tenant* and *infrastructure* organizations
6. Configure cobbler
    - Configure cobbler’s management of DHCP
    - Configure cobbler’s management of DNS
    - Configure cobbler’s management of PXE
7. Provision MGMT-1 node from Satellite
8. Migrate the Satellite VM from temporary node to MGMT-1
9. Provision additional cloud infrastructure management VM services on MGMT-1
    - Windows VM: RHEV-M
    - RHEL VM: JON
    - RHEL VM: MRG Manager

    - NFS service

10. Provision MGMT-2 node using temporary node from Satellite
11. Create file system management services in MGMT-2
    - NFS service based on ext3 file system
    - GFS2 file system
12. Make MGMT-1 and MGMT-2 RHCS cluster nodes including infrastructure management services
13. Install initial RHEV-H and RHEL/KVM hosts
14. Configure RHEV-M
    - Software
    - RHEV data center
    - RHEV Hosts

6.1 Install Management Build-Up System

A blade that is later used as a management cluster node is temporally installed to support the Satellite server. Once installed, the Satellite VM is used to provision the management cluster nodes.

The profile assigned to this blade has a 10G public network and two 8G fibre channel connections.

6.1.1 Common Scripts and Utilities

There are several scripts used throughout the build up procedure.

1. *varDefs.sh* - This script contains variable definitions used in many of the scripts. Defining these variable in one place provided flexibility in the modification of various specific settings during the process.

```
#!/bin/bash
FQD=cloud.lab.eng.bos.redhat.com
ILO_PW=24^goldA
JON_IP=10.16.136.45
JON_KS=jon.ks
JON_MAC=52:54:00:C0:DE:11
JON_NAME=jon-vm.cloud.lab.eng.bos.redhat.com
JON_PROFILE=jonProfile
LOGIN=Administrator
MGMT_DISTRO=ks-rhel-x86_64-server-5-u5
MGMT_KS=mgmt1.ks
MGMT_PROFILE=mgmtNode
MGMT1_ILO_PW=24^goldA
MGMT1_ILO=10.16.136.236
MGMT1_IP=10.16.136.10
MGMT1_IC=mgmt1-ic
```

```
MGMT1_ICIP=192.168.136.10
MGMT1_FC=mgmt_node1
MGMT1_MAC=00:17:A4:77:24:00
MGMT1_NAME=mgmt1.cloud.lab.eng.bos.redhat.com
MGMT1_PW=24^gold
MGMT2_ILO=10.16.136.232
MGMT2_ILO_PW=24^goldA
MGMT2_IP=10.16.136.15
MGMT2_IC=mgmt2-ic
MGMT2_ICIP=192.168.136.15
MGMT2_FC=mgmt_node2
MGMT2_MAC=00:17:A4:77:24:04
MGMT2_NAME=mgmt2.cloud.lab.eng.bos.redhat.com
MGMT2_PW=24^gold
MRGGRID_IP=10.16.136.50
MRGGRID_KS=mrggrid.ks
MRGGRID_MAC=52:54:00:C0:DE:22
MRGGRID_NAME=mrg-vm.cloud.lab.eng.bos.redhat.com
MRGGRID_PROFILE=mrgGridProfile
MSA_IP=10.16.136.248
MSA_USER=manage
MSA_PW=!manage
MASK=255.255.248.0
OA_IP=10.16.136.255
OA_PW=24^gold
RHELH01_VC_PROFILE=rhelh-01
RHELH_PROFILE=rhelHost:2:infrastructure
RHEVH_PROFILE=rhevh
RHEVH01_VC_PROFILE=rhevh-01
RHEVM_IP=10.16.136.40
RHEVM_NAME=rhevm-vm
SAT_FQDN=sat-vm.cloud.lab.eng.bos.redhat.com
SAT_INFRA_USER=infra
SAT_INFRA_PW=24^gold
SAT_IP=10.16.136.1
SAT_NAME=sat-vm
SAT_TENANT_USER=tenant
SAT_TENANT_PW=24^gold
VCM_IP=10.16.136.229
VCM_PW=24^gold
```

2. Several python based expect scripts were developed to communicate with vendor specific components. These scripts can be obtained from *http://people.redhat.com/~mlamouri* and provide the following commands allowing non-interactive secure shell communication with the target:
    - ilocommand – Integrated lights out
    - oacommand – On-Board Administrator
    - sacommand – Storage Array
    - vcmcommand – Virtual Connection Manager

3. A set of python based XMLRPC scripts were developed for remote communication with the ricci cluster daemon and can also be obtained from *http://people.redhat.com/~mlamouri/*:
    - riccicmd – generalized ricci communication, various commands can be supplied
    - addnfsexport – update cluster configuration adding NFS client resources
    - delnfsexport – update cluster configuration removing NFS client resource:

6.1.2 Prepare the Build-Up Installation

The following procedure was used to create the custom media to install the build-up system. The customization included configuring a default kickstart for the media and creating a *resources* directory containing files to be used in the configuration.

1. Download the released Red Hat Enterprise Linux 5.5 DVD via the RHN home page
    - Select *Download Software* on the left side of the page
    - Select the link for *Red Hat Enterprise Linux (v. 5 for 64-bit x86_64)*
    - Select the *Binary DVD (Server Core/Cluster/Cluster Storage/Virtualization)* link

2. Duplicate the media

    a) Create a directory for mounting the media

    ```
    mkdir -p /pub/projects/cloud/resources/wa/mnt
    ```

    b) Mount the media

    ```
    mount -o loop /pub/kits/rhel-server-5.5-x86_64-dvd.iso \
      /pub/projects/cloud/resources/wa/mnt
    ```

    c) Create directory for copying the media

    ```
    mkdir -p /pub/projects/cloud/resources/wa/r55
    ```

    d) Copy the expanded media

    ```
    cd /pub/projects/cloud/resources/wa/mnt; tar -cf - . | \
      (cd /pub/projects/cloud/resources/wa/r55; tar -xvpf -)
    ```

    e) Unmount the original media

    ```
    umount /pub/projects/cloud/resources/wa/mnt
    ```

3. Modify the duplicate tree

    a) Create the kickstart file */pub/projects/cloud/resources/wa/r55/ks.cfg* that:
    - installs and configures the system
    - copies the entire media to the installed system
    - sets the time and prepare NTP to start
    - registers the machine with RHN
    - prepares the NFS export of the installation media and customization
    - configures iptables (firewall)
    - converts the public network to a bridge named *cloud0*
    - disables LRO
    - prepares the actions to be performed on next boot

ks.cfg

```
install
cdrom
key <Installation Number>
lang en_US.UTF-8
keyboard us
#xconfig --startxonboot
skipx
network --device eth0 --bootproto static --ip 10.16.136.15 --netmask 255.255.248.0 --gateway
10.16.143.254 --nameserver 10.16.136.1,10.16.255.2 --hostname
mgmt2.cloud.lab.eng.bos.redhat.com
network --device eth1 --bootproto static --ip 192.168.136.15 --netmask 255.255.255.0 --hostname
mgmt2-ic
network --device eth2 --onboot no --bootproto dhcp --hostname
mgmt2.cloud.eng.lab.bos.redhat.com
network --device eth3 --onboot no --bootproto dhcp --hostname
mgmt2.cloud.eng.lab.bos.redhat.com
network --device eth4 --onboot no --bootproto dhcp --hostname
mgmt2.cloud.eng.lab.bos.redhat.com
network --device eth5 --onboot no --bootproto dhcp --hostname
mgmt2.cloud.eng.lab.bos.redhat.com
network --device eth6 --onboot no --bootproto dhcp --hostname
mgmt2.cloud.eng.lab.bos.redhat.com
network --device eth7 --onboot no --bootproto dhcp --hostname
mgmt2.cloud.eng.lab.bos.redhat.com
rootpw --iscrypted $1$1o751Xnc$kmQKHj6gtZ50IILNkHkkF0
firewall --enabled --port=22:tcp
authconfig --enableshadow --enablemd5
selinux --permissive
reboot
services --enabled=multipathd
timezone --utc America/New_York
bootloader --location=mbr --driveorder=cciss/c0d0 --append="rhgb quiet console=ttyS0,115200"
# The following is the partition information you requested
# Note that any partitions you deleted are not expressed
# here so unless you clear all partitions first, this is
# not guaranteed to work
driver scsi cciss
clearpart --all --drives=cciss/c0d0
part /boot --fstype ext3 --size=200 --ondisk=cciss/c0d0
part pv.01 --size=0 --grow --ondisk=cciss/c0d0
volgroup InstallVG --pesize=32768 pv.01
logvol swap --fstype swap --name=SwapLV --vgname=InstallVG --size=1024 --grow
--maxsize=10240
logvol / --fstype ext4 --name=RootLV --vgname=InstallVG --size=1024 --grow

%packages
@admin-tools
@base
@core
```

```
@editors
@graphical-internet
@graphics
@java
@kvm
@legacy-software-support
@text-internet
@base-x
kexec-tools
iscsi-initiator-utils
bridge-utils
fipscheck
device-mapper-multipath
sgpio
emacs
libsane-hpaio
xorg-x11-utils
xorg-x11-server-Xnest
sg3_utils
ntp
pexpect

%pre
#Save some information about the cdrom device
ls -l /tmp/cdrom > /tmp/cdrom.ls

%post --nochroot
(
#If the device is no longer there by the time post starts, create it
if [ ! -b /tmp/cdrom ]
then
  #get the major number
  major=$(cat /tmp/cdrom.ls | cut -d" " -f5)
  #strip of the trailing comma
  major=${major%,}
  #get the minor number
  minor=$(cat /tmp/cdrom.ls | cut -d" " -f6)
  #make sure we have what we need; create device node if so
  [ -n "$major" -a -n "$minor" ] && mknod /tmp/cdrom b $major $minor
fi

#make sure mount directory exists
if [ ! -d /mnt/sysimage/root/distro ]
then
  mkdir -p /mnt/sysimage/root/distro
fi

#should be ready to mount the optical media
mount -t iso9660 -o ro /tmp/cdrom /mnt/sysimage/root/distro
# mount -t iso9660 -o ro /tmp/cdrom /mnt/source
# # copy the entire content onto the created machine
# mkdir /mnt/sysimage/root/distro
# (cd /mnt/source; tar -cf - .  ) | (cd /mnt/sysimage/root/distro; tar -xpf -)

) 2>&1 | tee /mnt/sysimage/root/ks_post.log

%post
(

#set the time from a server, then use this to set the hwclock. Enable ntp
/usr/sbin/ntpdate -bu 10.16.255.2
hwclock --systohc
chkconfig ntpd on
cat <<EOF>/etc/ntp.conf
restrict default kod nomodify notrap nopeer noquery
restrict 127.0.0.1
driftfile /var/lib/ntp/drift
keys /etc/ntp/keys
server 10.16.255.2
server 10.16.255.3
EOF

#register the machine with RHN so latest versions can be made available
rhnreg_ks --profilename=mgmt2.cloud.lab.eng.bos.redhat.com --username=milo
--password=<password>

#prepare to export the software distribution to install the satellite system
echo "/root/distro          *(ro)" > /etc/exports
chkconfig nfs on

#prepare iptables
/bin/cp /etc/sysconfig/iptables /tmp/iptables
/usr/bin/head -n -2 /tmp/iptables > /etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -m physdev --physdev-is-bridged -j ACCEPT" >>
/etc/sysconfig/iptables
/usr/bin/tail -2 /tmp/iptables >> /etc/sysconfig/iptables

# Convert the public network in a bridge named cloud0
/root/distro/resources/cvt_br.sh eth0 cloud0

# Put a reasonable resolv.conf in place
#/bin/cp /etc/resolv.conf /etc/resolv.conf.orig
#/bin/cp /root/distro/resources/resolv.conf /etc/resolv.conf

#disable LRO - BZ 518531
echo "options bnx2x disable_tpa=1" >> /etc/modprobe.conf

# prepare actions to be performed on next boot.
/bin/cp /etc/rc.d/rc.local /etc/rc.d/rc.local.shipped
```

```
cat /root/distro/resources/temp.rc.local.add >> /etc/rc.d/rc.local

#update to latest software
yum -y update
) 2>&1 | tee /root/ks_post2.out
```

b) The *cvt_br.sh* script, called from the above kickstart, is used to convert an existing network interface into a bridge

```
#!/bin/bash

# if 2 parameters were not passed in, display usage
if [[ $# -ne 2 ]]
then
   echo "Usage - $0 NIC <bridge>"
   exit
fi

#Verify source NIC exists
if [[ ! -f /etc/sysconfig/network-scripts/ifcfg-$1 ]]
then
   echo "Error source: /etc/sysconfig/network-scripts/ifcfg-$1 does not exist!"
   exit
fi

#Confirm bridge doesn't exists
if [[ -f /etc/sysconfig/network-scripts/ifcfg-$2 ]]
then
   echo "Error bridge: /etc/sysconfig/network-scripts/ifcfg-$2 exists!"
   exit
fi

#Copy existing NIC network specific info to the bridge file, then append DEVICE, TYPE, and ONBOOT
grep -e BOOTPROTO -e IPADDR -e NETMASK -e GATEWAY /etc/sysconfig/network-scripts/ifcfg-$1 > /etc/sysconfig/network-scripts/ifcfg-$2
echo "DEVICE=$2" >> /etc/sysconfig/network-scripts/ifcfg-$2
echo "TYPE=Bridge" >> /etc/sysconfig/network-scripts/ifcfg-$2
echo "ONBOOT=yes" >> /etc/sysconfig/network-scripts/ifcfg-$2

#Remove the network specific info from the NIC and add BRIDGE, BOOTPROTO
mv /etc/sysconfig/network-scripts/ifcfg-$1 /etc/sysconfig/network-scripts/ifcfg-$1.bak
grep -v -e BOOTPROTO -e IPADDR -e NETMASK -e GATEWAY -e ONBOOT /etc/sysconfig/network-scripts/ifcfg-$1.bak > /etc/sysconfig/network-scripts/ifcfg-$1
echo "BRIDGE=$2" >> /etc/sysconfig/network-scripts/ifcfg-$1
echo "BOOTPROTO=none" >> /etc/sysconfig/network-scripts/ifcfg-$1
```

c) *temp.rc.local.add*, also called from the kickstart created in the above step 3a, contains actions that are executed on the next boot

```
(
#mount the media
/bin/mount -o ro /dev/scd0 /root/distro

# source env vars
if [[ -x varDefs.sh ]] ; then
  source varDefs.sh
elif [[ -x /root/varDefs.sh ]] ; then
  source /root/varDefs.sh
elif [[ -x /root/resources/varDefs.sh ]] ; then
  source /root/resources/varDefs.sh
elif [[ -x /root/distro/resources/varDefs.sh ]] ; then
  source /root/distro/resources/varDefs.sh
else
  echo "didn't find a varDefs.sh file"
fi

# install ssh pexpect commands
/root/distro/resources/instpyCmds.sh /root/distro/resources

# configure storage volumes, presentation, LVM volumes
/root/distro/resources/prep_storage.sh

# generate ssh configuration to be passed to other cluster members
/root/distro/resources/prep_ssh.sh

# Create satellite
/root/distro/resources/create_satVM.sh

# remove the temporary node's RHN registration
/root/distro/resources/wipeRHNReg.py ${MGMT2_NAME}

mv /etc/rc.d/rc.local.shipped /etc/rc.d/rc.local

)  >> /var/log/rc.local.out 2>&1
```

d) Modify the *isolinux/isolinux.cfg* file to use the *ks.cfg* that exists on the new media

```
default ks
prompt 1
timeout 60
display boot.msg
F1 boot.msg
F2 options.msg
F3 general.msg
F4 param.msg
F5 rescue.msg
label linux
  kernel vmlinuz
  append initrd=initrd.img
label text
  kernel vmlinuz
  append initrd=initrd.img text
```

```
label ks
  kernel vmlinuz
  append ks=cdrom:/ks.cfg initrd=initrd.img console=ttyS0,115200 nostorage
label local
  localboot 1
label memtest86
  kernel memtest
  append -
```

e) Create and populate a *resources* subdirectory with files for later use. Many of the files found in this directory are detailed in this and following sections of this document including the appendixes.

4. Call the *mkMedia* script to create an ISO image using the modified directory

```
#!/bin/bash
mkisofs -r -N -L -d -J -T -b isolinux/isolinux.bin -c isolinux/boot.cat -no-emul-boot -V "RHCF" -boot-
load-size 4 -boot-info-table  -o /pub/projects/cloud/resources/wa/rhcf.iso
/pub/projects/cloud/resources/wa/r55
which implantisomd5 > /dev/null 2>/dev/null
if [[ $? ]]
then
  if [[ -x /usr/lib/anaconda-runtime/implantisomd5 ]]
  then
    /usr/lib/anaconda-runtime/implantisomd5 rhcf.iso
  fi
else
  implantisomd5 rhcf.iso
fi
```

6.1.3 Install

With due attention to the above preparations, the actual installation is quick and easy. The assumed starting state is that the blade is powered off.

1. Call the following script which unmounts any previously mounted virtual CDROM, mounts the previously created media, sets the boot order so CDROM is first, powers on the blade, waits for the installation to initiate, and changes the boot order so the hard drive boots first. By default, the CDROM remains mapped throughout reboots.

```
#!/bin/bash -x

# source env vars
if [[ -x varDefs.sh ]] ; then
  source varDefs.sh
elif [[ -x /pub/projects/cloud/resources/wa/r55/resources/varDefs.sh ]] ; then
  source /pub/projects/cloud/resources/wa/r55/resources/varDefs.sh
else
  echo "didn't find a varDefs.sh file"
fi

ilocommand --ilourl //${LOGIN}:${MGMT2_ILO_PW}@${MGMT2_ILO} vm cdrom set disconnect
sleep 2
```

```
ilocommand --ilourl //${LOGIN}:${MGMT2_ILO_PW}@${MGMT2_ILO} set
/map1/oemhp_vm1/cddr1
oemhp_image=http://irish.lab.bos.redhat.com/pub/projects/cloud/resources/wa/rhcf.iso
ilocommand --ilourl //${LOGIN}:${MGMT2_ILO_PW}@${MGMT2_ILO} vm cdrom set connect

while [[ ! `ilocommand -i //${LOGIN}:${MGMT2_ILO_PW}@${MGMT2_ILO} set
/system1/bootconfig1/bootsource1 bootorder=1 | grep status=0` ]]; do sleep 20; done

ilocommand --ilourl //${LOGIN}:${MGMT2_ILO_PW}@${MGMT2_ILO} power on

sleep 600
while [[ ! `ilocommand -i //${LOGIN}:${MGMT2_ILO_PW}@${MGMT2_ILO} set
/system1/bootconfig1/bootsource3 bootorder=1 | grep status=0` ]] ; do sleep 20 ; done
```

2. The system installs.

6.2 Create Satellite System

6.2.1 Prepare Host

As mentioned in Section **6.1.2** , the final action of the kickstart is to configure a script that is called at first boot. This script's actions are executed in the following order:

1. The customized *rc.local* file mounts the CDROM

```
 [...]
/sbin/mount -o ro /dev/scd0 /root/distro
 [...]
```

2. *instpyCmds.sh* - installs python based expect commands

```
#!/bin/bash
#
# This script installs the CLI command libraries for ILO, Storage Array,
# Virtual Console, and Onboard Administrator

#If an argument was passed, change to that directory
if [[ $# -gt 0 ]]
then
  cd $1
fi

#find the rpm file names
SAFILE=`ls salib-*.noarch.rpm`
VCMFILE=`ls vcmlib-*.noarch.rpm`
ILOFILE=`ls ilolib-*.noarch.rpm`
OAFILE=`ls oalib-*.noarch.rpm`

#calculate the number of RPMs for each tool
SANUM=`echo $SAFILE | wc -w`
```

```
VCMNUM=`echo $VCMFILE | wc -w`
ILONUM=`echo $ILOFILE | wc -w`
OANUM=`echo $OAFILE | wc -w`

#install storage array command tool
if [[ $SANUM -eq 0 ]]
then
  echo "No salib source!"
elif [[ $SANUM -gt 1 ]]
then
 echo "Too man salib source files!"
else
  yum -y --nogpgcheck localinstall ${SAFILE}
fi

#install Virtual Connect Manager command tool
if [[ $VCMNUM -eq 0 ]]
then
  echo "No vcmlib source!"
elif [[ $VCMNUM -gt 1 ]]
then
 echo "Too man vcmlib source files!"
else
  yum -y --nogpgcheck localinstall ${VCMFILE}
fi

#install Onboard Administrator command tool
if [[ $OANUM -eq 0 ]]
then
  echo "No oalib source!"
elif [[ $OANUM -gt 1 ]]
then
 echo "Too man oalib source files!"
else
  yum -y --nogpgcheck localinstall ${OAFILE}
fi

#install ILO (integrated Lights Out) command tool
if [[ $ILONUM -eq 0 ]]
then
  echo "No ilolib source!"
elif [[ $ILONUM -gt 1 ]]
then
 echo "Too man ilolib source files!"
else
  yum -y --nogpgcheck localinstall ${ILOFILE}
fi
```

3. *prep_storage.sh*
    - assigns host HBA aliases on the storage array

- creates and presents volumes from storage array
- configures system to access presented volumes
- configures LVM group to be used with management services

```
#!/bin/bash

#source variables
if [[ -x varDefs.sh ]]
then
  source varDefs.sh
elif [[ -x /root/varDefs.sh ]]
then
  source /root/varDefs.sh
elif [[ -x /root/resources/varDefs.sh ]]
then
  source /root/resources/varDefs.sh
elif [[ -x /root/distro/resources/varDefs.sh ]]
then
  source /root/distro/resources/varDefs.sh
else
  echo "didn't find a varDefs.sh file"
fi

#Acquire short hostname
SHORTHOST=`hostname --short`

#Add the HBAs of this host to the MSA storage array
/root/distro/resources/map_fc_aliases.sh

# Create storage LUNs for the MgmtServices volume group,
# GFS2 storage of VM config files, and the RHEV-M VM image disk
sacommand --saurl //${MSA_USER}:${MSA_PW}@${MSA_IP} create volume MgmtServices vdisk
VD1 size 1TB lun 1
sacommand --saurl //${MSA_USER}:${MSA_PW}@${MSA_IP} create volume GFS2 vdisk VD3
size 50GB lun 3
sacommand --saurl //${MSA_USER}:${MSA_PW}@${MSA_IP} create volume RHEVStorage1
vdisk VD4 size 1TB lun 4

#Present the MgmtServices and GFS2 storage volumes to each
#HBA in this host
for f in `sacommand --saurl //${MSA_USER}:${MSA_PW}@${MSA_IP} show hosts | grep $
{SHORTHOST}_ | awk '{print $2}'`
do
  sacommand --saurl //${MSA_USER}:${MSA_PW}@${MSA_IP} map volume MgmtServices
access rw ports a1,a2 lun 1 host ${f}
  sacommand --saurl //${MSA_USER}:${MSA_PW}@${MSA_IP} map volume GFS2 access rw
ports a1,a2 lun 2 host ${f}
done

#Unpresent the same LUNs to all other host known to the array
```

```
sacommand --saurl //${MSA_USER}:${MSA_PW}@${MSA_IP} unmap volume MgmtServices
sacommand --saurl //${MSA_USER}:${MSA_PW}@${MSA_IP} unmap volume GFS2
sacommand --saurl //${MSA_USER}:${MSA_PW}@${MSA_IP} unmap volume RHEVStorage1

#Rescan the SCSI bus to discover newly presented LUNs
/usr/bin/rescan-scsi-bus.sh

#Deploy multipath configuration file preconfigured with recommended
#MSA array settings for optimal performance
/bin/cp /root/distro/resources/multipath.conf.template /etc/multipath.conf

#Build multipath aliases stanza to add to the configuration file
/root/distro/resources/buildMpathAliases.sh ${MSA_IP} >> /etc/multipath.conf

#Reload multipath configuration file
service multipathd reload

#Create the Management services volume group
pvcreate /dev/mapper/MgmtServices_disk
vgcreate MgmtServicesVG /dev/mapper/MgmtServices_disk

#Create the logical volume for the satellite server
lvcreate -n SatVMvol -L 300G MgmtServicesVG
```

a) The above *prep_storage.sh* script calls two others. The first, *map_fc_aliases.sh*, defines host aliases for the systems World Wide Names for the system's fibre channel ports on the storage array.

```
#!/bin/bash
#
# This script will add the WWN of any HBA in the executing host to
# the MSA storage array

# source env vars
if [[ -x varDefs.sh ]] ; then
  source varDefs.sh
elif [[ -x /root/varDefs.sh ]] ; then
  source /root/varDefs.sh
elif [[ -x /root/resources/varDefs.sh ]] ; then
  source /root/resources/varDefs.sh
elif [[ -x /root/distro/resources/varDefs.sh ]] ; then
  source /root/distro/resources/varDefs.sh
else
  echo "didn't find a varDefs.sh file"
fi

SHORTHOST=`hostname --short`
if [[ -d /sys/class/fc_host ]]
then
  NUM=`sacommand --saurl //${MSA_USER}:${MSA_PW}@${MSA_IP} show hosts | grep $
{SHORTHOST}_ | wc -l`
```

```
  if [[ $NUM -gt 0 ]]
  then
    echo "Aliases for this host already exists!"
  else

    cd /sys/class/fc_host
    for f in host*
    do
      WWN=`cat ${f}/port_name | cut -f 2 -d 'x'`
      sacommand --saurl //${MSA_USER}:${MSA_PW}@${MSA_IP} set host-name id ${WWN}
${SHORTHOST}_${f}
    done
  fi
fi
```

b) The second script, *buildMpathAliases.sh,* also called by the storage preparation script, searches the multipath devices and assembles a portion of the multipath configuration file that provides aliases for fibre channel devices.

```
#!/bin/bash
# This script will talk to the specified array and retrieve all existing volume and use
# the name of the volume to create an appropriate multipaths stanza
#    developed to work with MSA2000s, unsure if any others would work

# usage: ./buildMpathAliases.sh <storage_array_name>

# source env vars
if [[ -x varDefs.sh ]] ; then
  source varDefs.sh
elif [[ -x /root/varDefs.sh ]] ; then
  source /root/varDefs.sh
elif [[ -x /root/resources/varDefs.sh ]] ; then
  source /root/resources/varDefs.sh
elif [[ -x /root/distro/resources/varDefs.sh ]] ; then
  source /root/distro/resources/varDefs.sh
else
  echo "didn't find a varDefs.sh file"
fi

#check that an argument was passed
if [ $# -ne 1 ] ; then
  echo "usage: buildMpathAliases.sh <storage_array_name>"
  exit 1
fi
array=$1

#grab the WWN for all the local disks
for i in `awk '/sd/ {print $4}' /proc/partitions` ; do
  echo "`scsi_id -g -u -s /block/$i`" >> /tmp/devs1
done
sort -u /tmp/devs1 > /tmp/devs2
```

```
#get the data from the array, assumes all virtual disk have VD the name
sacommand --saurl //${MSA_USER}:${MSA_PW}@${array} "show volumes" | grep VD >
/tmp/vols

#begin the statement construction, the WWN do not line up, so only sections are compared
echo "multipaths {"
cat /tmp/vols | while read line ; do
  alias=`echo $line | cut -d ' ' -f 2 | cut -c7-`
  temp=`echo $line | cut -d ' ' -f 4 | cut -c 8-12``echo $line | cut -d ' ' -f 4 | cut -c 17-23`
  wwid=`grep $temp /tmp/devs2 | sort -u`
  if [ "$wwid" != "" ]; then
    echo -e "\tmultipath {\n\t\twwid\t\t$wwid\n\t\talias\t\t${alias}_disk\n\t}"
  fi
done
echo "}"

#cleanup
rm -f /tmp/devs? /tmp/vols
```

4. *prep_ssh.sh* – generates secure shell keys and configuration files, and temporarily places these files on the GFS2 volume for passing to the other cluster member

```
#!/bin/bash
# This script will create a FS for temporary use on the disk to be
# used for GFS2, generate ssh config and copy the ssh config to this disk

#Create ext2 file system
mkfs /dev/mapper/GFS2_disk

#Mount temp file system
mount /dev/mapper/GFS2_disk /mnt

#Create the .ssh dir and generate individual and cluster RSA key pairs
mkdir /root/.ssh
chmod 700 /root/.ssh
ssh-keygen -q -t rsa -N "" -f /root/.ssh/id_rsa

#Add the public key to the authorized keys
/bin/cp /root/.ssh/id_rsa.pub /root/.ssh/authorized_keys

#Add cluster key and allow ssh to automatically add all hosts to the
#host file as they connect
echo StrictHostKeyChecking no >> /root/.ssh/config

#Copy the ssh configuration to the temp file system and unmount
/bin/cp /root/.ssh/id_rsa /root/.ssh/id_rsa.pub  /root/.ssh/authorized_keys /root/.ssh/config /mnt/
umount /mnt
```

5. *create_satVM.sh* – creates the satellite VM, detailed in the next section

```
#!/bin/bash
```

```
# source env vars
if [[ -x varDefs.sh ]] ; then
  source varDefs.sh
elif [[ -x /root/varDefs.sh ]] ; then
  source /root/varDefs.sh
elif [[ -x /root/resources/varDefs.sh ]] ; then
  source /root/resources/varDefs.sh
elif [[ -x /root/distro/resources/varDefs.sh ]] ; then
  source /root/distro/resources/varDefs.sh
else
  echo "didn't find a varDefs.sh file"
fi

#Temporarily allow host to get all packets from the VM (allowing NFS traffic)
iptables -I RH-Firewall-1-INPUT -s $SAT_IP -j ACCEPT

#Create the VM
virt-install -n $SAT_NAME -r 8192 --vcpus=4 --cpuset=auto --os-type=linux --os-variant=rhel5.4
--accelerate -w bridge:cloud0 --vnc -l nfs:${MGMT2_IP}:/root/distro --disk
path=/dev/MgmtServicesVG/SatVMvol,bus=virtio -x "ks=nfs:$
{MGMT2_IP}:/root/distro/resources/sat.ks.cfg ip=$SAT_IP gateway=$OA_IP netmask=$MASK"
--noautoconsole --wait=-1

#Restart iptables
service iptables restart
```

6. *wipeRHNReg.py* – remove the RHN registration for the specified system. Because the system is reinstalled and registered to satellite, the RHN registration is not required.

```
#!/usr/bin/python
"""
remove the registration of the system passed as the only parameter
"""

import xmlrpclib
import sys

SATELLITE_URL = "https://rhn.redhat.com/rpc/api"
SATELLITE_LOGIN = "<UserName>"
SATELLITE_PASSWORD = "<Password>"

def main():

    if len(sys.argv) < 2:
        print "Usage: ",sys.argv[0],"systemName"
        sys.exit(-2);

    # retrieve the passed parameter
    deleteName = sys.argv[1]

    #open connection and log in
    client = xmlrpclib.Server(SATELLITE_URL, verbose=0)
    key = client.auth.login(SATELLITE_LOGIN, SATELLITE_PASSWORD)

    #retrieve all the systems
    AllSystems = client.system.listUserSystems(key)

    # Initialize the list IDs of systems that match the id
    IDs = []

    #Loop through all the systems, if the name matches save the Id
    for Sys in AllSystems:
        if Sys['name'] == deleteName:
            IDs.append(int(Sys['id']))

    # Either print the system name is no systems matched, or delete all matched entries
    if IDs == []:
        print "No registered systems matched name: ", deleteName
    else:
        client.system.deleteSystems(key,IDs)

    client.auth.logout(key)

if __name__ == "__main__":
    sys.exit(main())
```

6.2.2 Install VM for Satellite

1. The kickstart configuration file used to install the satellite system is shown. In addition to installing the systems, the post script:
    - mounts the CDROM and copies the *resources* directory to the system
    - sets the time, prepares NTP
    - configures iptables/firewalls
    - registers the system with RHN
    - updates software
    - configures DHCP and DNS
    - performs some cobbler related SELinux configurations
    - deploys a *resolv.conf* file
    - prepares actions to be performed on next boot

    sat.ks.cfg

```
install
nfs --server=10.16.136.15 --dir=/root/distro
key <Installation Number>
lang en_US.UTF-8
keyboard us
#xconfig --startxonboot
skipx
network --device eth0 --bootproto static --ip 10.16.136.1 --netmask 255.255.248.0 --gateway
```

```
10.16.143.254 --nameserver 10.16.136.1,10.16.255.2 --hostname sat-vm.cloud.lab.eng.bos.redhat.com
rootpw --iscrypted $1$1o751Xnc$kmQKHj6gtZ50IILNkHkkF0
firewall --enabled --port=22:tcp
authconfig --enableshadow --enablemd5
selinux --permissive
reboot
timezone --utc America/New_York
bootloader --location=mbr --driveorder=vda --append="rhgb quiet"
# The following is the partition information you requested
# Note that any partitions you deleted are not expressed
# here so unless you clear all partitions first, this is
# not guaranteed to work
clearpart --all --initlabel
part /boot --fstype ext3 --size=200
part pv.01 --size=0 --grow
volgroup SatVG --pesize=32768 pv.01
logvol swap --fstype swap --name=SwapLV --vgname=SatVG --size=1024 --grow --maxsize=10240
logvol / --fstype ext4 --name=RootLV --vgname=SatVG --size=1024 --grow

%packages
@admin-tools
@base
@core
@editors
@graphics
@java
@text-internet
@base-x
kexec-tools
emacs
bind
system-config-bind
dhcp
ntp
yum-utils
pexpect

%post --nochroot
(
mkdir /mnt/source
mount 10.16.136.15:/root/distro /mnt/source
cp -r /mnt/source/resources /mnt/sysimage/root/
) 2>&1 | tee /mnt/sysimage/root/ks_post.log

%post
(
# Set time and turn on ntp for the next
/usr/sbin/ntpdate -bu 10.16.255.2
chkconfig ntpd on
/bin/cat <<EOF>/etc/ntp.conf
```

```
restrict default kod nomodify notrap nopeer noquery
restrict 127.0.0.1
driftfile /var/lib/ntp/drift
keys /etc/ntp/keys
server 10.16.136.10
server 10.16.136.15
EOF

# Configure iptables
/bin/cp /etc/sysconfig/iptables /tmp/iptables
/usr/bin/head -n -2 /tmp/iptables > /etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp -m state --state NEW -m tcp --dport 53 -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p udp -m state --state NEW -m udp --dport 53 -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp -m state --state NEW -m tcp --dport 68 -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p udp -m state --state NEW -m udp --dport 68 -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p udp -m state --state NEW -m udp --dport 69 -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp -m state --state NEW -m tcp --dport 69 -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p udp -m udp --dport 80 -j ACCEPT" >> /etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 80 -j ACCEPT" >> /etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p udp -m udp --dport 443 -j ACCEPT" >> /etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 443 -j ACCEPT" >> /etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 4545 -j ACCEPT" >> /etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p udp -m udp --dport 4545 -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 5222 -j ACCEPT" >> /etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p udp -m udp --dport 5222 -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p udp -m state --state NEW -m udp --dport 25150 -j ACCEPT"
>> /etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp -m state --state NEW -m tcp --dport 25151 -j ACCEPT" >>
/etc/sysconfig/iptables
/usr/bin/tail -2 /tmp/iptables >> /etc/sysconfig/iptables

#register stem with RHN hosted
rhnreg_ks --profilename=sat-vm.cloud.lab.eng.bos.redhat.com --username=milo
--password=<password> --subscription=a9b1fb74a9ca05fb

# update software
yum -y update

#configure DHCP
/bin/cp /root/resources/dhcpd.conf /etc/dhcpd.conf
chkconfig dhcpd on
```

```
# configure DNS
/bin/cp /root/resources/db.* /var/named/
/bin/cp /root/resources/named.conf /etc/
chkconfig named on

# cobbler preparation
/usr/sbin/semanage fcontext -a -t public_content_t "/var/lib/tftpboot/.*"
/usr/sbin/semanage fcontext -a -t public_content_t "/var/www/cobbler/images/.*"
setsebool -P httpd_can_network_connect true

# update DNS resolution
/bin/cp /etc/resolv.conf /etc/resolv.conf.orig
/bin/cp /root/resources/resolv.conf /etc/resolv.conf

#prepared actions to be performed on net boot
/bin/cp /etc/rc.d/rc.local /etc/rc.d/rc.local.shipped
cat /root/resources/sat.rc.local.add >> /etc/rc.d/rc.local
) 2>&1 | tee /root/satprep.log
```

a) This script performs actions on the first reboot of the VM

sat.rc.local.add

```
# install the ssh expect scripts
date >> /var/log/rc.local.out 2>&1
/root/resources/instpyCmds.sh /root/resources >> /var/log/rc.local.out 2>&1

#install and prime the ricci access commands/library
yum -y --nogpgcheck localinstall /root/resources/riccilib-current.noarch.rpm >> /var/log/rc.local.out
2>&1

#source variables
if [[ -x varDefs.sh ]]
then
  source varDefs.sh
elif [[ -x /root/varDefs.sh ]]
then
  source /root/varDefs.sh
elif [[ -x /root/resources/varDefs.sh ]]
then
  source /root/resources/varDefs.sh
elif [[ -x /root/distro/resources/varDefs.sh ]]
then
  source /root/distro/resources/varDefs.sh
else
  echo "didn't find a varDefs.sh file"
fi

# eject cd
ilocommand --ilourl //Administrator:24^goldA@10.16.136.232 vm cdrom set disconnect

# install satellite software
```

```
echo "-" >> /var/log/rc.local.out 2>&1
echo "- begin sat install" >> /var/log/rc.local.out 2>&1
echo "-" >> /var/log/rc.local.out 2>&1
date >> /var/log/rc.local.out 2>&1
/root/resources/instSat.sh  >> /var/log/rc.local.out 2>&1

# configure cobbler
echo "-" >> /var/log/rc.local.out 2>&1
echo "- begin cobbler config" >> /var/log/rc.local.out 2>&1
echo "-" >> /var/log/rc.local.out 2>&1
date >> /var/log/rc.local.out 2>&1
/root/resources/configCobbler.sh >> /var/log/rc.local.out 2>&1

#remove the satellite software media before making the resources available via the web
echo "-" >> /var/log/rc.local.out 2>&1
echo "- copy resource to web area" >> /var/log/rc.local.out 2>&1
echo "-" >> /var/log/rc.local.out 2>&1
date >> /var/log/rc.local.out 2>&1
/bin/rm /root/resources/redhat-rhn-satellite-5.3-server-x86_64-5-embedded-oracle.iso >>
/var/log/rc.local.out 2>&1
/bin/cp -r /root/resources /var/www/html/pub/  >> /var/log/rc.local.out 2>&1

#subscribe channels
echo "-" >> /var/log/rc.local.out 2>&1
echo "- start subscribing channels" >> /var/log/rc.local.out 2>&1
echo "-" >> /var/log/rc.local.out 2>&1
date >> /var/log/rc.local.out 2>&1
/root/resources/subChannels.sh >> /var/log/rc.local.out 2>&1

# synchronize the channels that were previously described, use local satellite first for speed
purposes
echo "-" >> /var/log/rc.local.out 2>&1
echo "- downloading sat channels" >> /var/log/rc.local.out 2>&1
echo "-" >> /var/log/rc.local.out 2>&1
date >> /var/log/rc.local.out 2>&1
mkdir /pub > /var/log/satSync 2>&1

#either /pub should be mounted or the certificate must be retrieved
echo "irish.lab.bos.redhat.com:/pub /pub nfs defaults 0 0" >> /etc/fstab
/bin/mount /pub >> /var/log/satSync 2>&1
#wget http://irish.lab.bos.redhat.com/pub/RHN-ORG-TRUSTED-SSL-CERT -O /pub/RHN-ORG-
TRUSTED-SSL-CERT >> /var/log/satSync 2>&1 &

#These next two line attempt a direct import
satellite-sync --mount-point=/pub/projects/cloud/resources/wa/satDump -c rhel-x86_64-server-5 -c
rhel-x86_64-server-vt-5 -c rhel-x86_64-server-5-mrg-grid-1 -c rhel-x86_64-server-5-mrg-
management-1 -c rhel-x86_64-server-cluster-5 -c rhn-tools-rhel-x86_64-server-5 -c rhel-x86_64-
server-cluster-storage-5 -c rhel-x86_64-rhev-mgmt-agent-5 -c rhel-x86_64-server-5-mrg-grid-
execute-1 -c rhel-x86_64-server-5-mrg-messaging-1  >> /var/log/satSync 2>&1
```

```
( nice -n -15 satellite-sync --iss-parent=irish.lab.bos.redhat.com --ca-cert=/pub/RHN-ORG-TRUSTED-SSL-CERT; nice -n -15 satellite-sync --iss-parent=irish.lab.bos.redhat.com --ca-cert=/pub/RHN-ORG-TRUSTED-SSL-CERT; nice -n -15 satellite-sync ) >> /var/log/satSync 2>&1

# prepare another set of actions to be performed on next boot (brought down from temp node then started on provisioned mgmt1)
echo "-" >> /var/log/rc.local.out 2>&1
echo "- Preparing next boot round of actions" >> /var/log/rc.local.out 2>&1
echo "-" >> /var/log/rc.local.out 2>&1
date >> /var/log/rc.local.out 2>&1
/bin/cat /etc/rc.d/rc.local.shipped /root/resources/sat.rc.local2.add > /etc/rc.d/rc.local
```

6.2.3 Install and Configure RHN Satellite Software

As mentioned, the final action of the satellite kickstart is to configure several scripts and commands to be executed on the first boot via */etc/rc.local*. The scripts and commands called by */etc/rc.local* are invoked in the order listed below. Note the italicized content below are commands excerpted from the *rc.local* file listed in the previous section.

> NOTE: Red Hat Bugzilla 593048 may occur during any of the VM installations on KVM hosts. The logs generated during this process must be checked to confirm success.

1. *instpyCmds.sh* – see Section **6.2.1**
2. Install the ricci communication utilities – see Section **6.1.1**

```
 [...]
yum -y --nogpgcheck localinstall /root/resources/riccilib-current.noarch.rpm >> /tmp/rc.local.out 2>&1
 [...]
```

3. *instSat.sh* - install RHN Satellite software via this script that:
    - mounts media
    - installs software
    - updates software
    - restarts satellite

```
#!/bin/bash
#
# This script install the satellite software using a preconfigured answers file

#prepare install media
mkdir /media/cdrom
mount -o loop /root/resources/redhat-rhn-satellite-5.3-server-x86_64-5-embedded-oracle.iso /media/cdrom

# install software
cd /media/cdrom
./install.pl --answer-file=/root/resources/answers.txt --non-interactive

#update software just installed and restart satellite to use this software
```

```
yum -y update
rhn-satellite restart
```

4. *configCobbler.sh* - configure cobbler using this script that:
   - performs recommended SELinux changes
   - updates settings
   - creates the template *multipath.conf.template*
   - creates named and zone templates
   - has cobbler create files from templates

```
#!/bin/bash
# This script will configure cobbler:

# perform recommended SELinux changes
/usr/sbin/semanage fcontext -a -t public_content_t "/var/lib/tftpboot/.*"
/usr/sbin/semanage fcontext -a -t public_content_t "/var/www/cobbler/images/.*"
setsebool -P httpd_can_network_connect true

#place preconfigured settings files into place
/bin/cp /root/resources/settings /etc/cobbler/settings

# Create the dhcpd.template from the existing dhcpd.conf
/usr/bin/head -n -1 /etc/dhcpd.conf > /etc/cobbler/dhcp.template
cat <<'EOF'>>/etc/cobbler/dhcp.template
#for dhcp_tag in $dhcp_tags.keys():
    ## group could be subnet if the dhcp tags align with the subnets
    ## or any valid dhcpd.conf construct ... if the default dhcp tag
    ## in cobbler is the only one used, the group block can be deleted for a
    ## flat configuration
## group for Cobbler DHCP tag: $dhcp_tag

        #for mac in $dhcp_tags[$dhcp_tag].keys():
                #set iface = $dhcp_tags[$dhcp_tag][$mac]
    host $iface.name {
        hardware ethernet $mac;
        #if $iface.ip_address:
        fixed-address $iface.ip_address;
        #end if
        #if $iface.hostname:
        option host-name "$iface.hostname";
        #end if
    }
        #end for
#end for
}
EOF

# Create the named.template from the existing named.conf
#   assumes that the zone configuration are each a single line
grep -v -e db.10.16. -e db.cloud /etc/named.conf > /etc/cobbler/named.template
```

```
cat <<'EOF'>>/etc/cobbler/named.template
#for $zone in $forward_zones
zone "${zone}." { type master; file "$zone"; };
#end for
#for $zone, $arpa in $reverse_zones
zone "${arpa}." { type master; file "$zone"; };
#end for
EOF

# Create the zone templated from the existing files
#  The "db." in the original zone file names is dropped
mkdir /etc/cobbler/zone_templates
(
cd /var/named
for f in db.cloud.lab.eng.bos.redhat.com db.10.16.*
do
  /bin/cp $f /etc/cobbler/zone_templates/${f#db.}
  echo \$host_record >> /etc/cobbler/zone_templates/${f#db.}
done
)
sed -i '/SOA/s/9876/\$serial/' /etc/cobbler/zone_templates/*

# Don't need to change tftp setting, default were good

#create all files from cobbler templates
cobbler sync
```

5. Copy resources (excluding RHN Satellite installation media) to an area accessible via the web server on the satellite system. A simple `satellite-sync` would be sufficient, possibly followed by another for verification, however the steps listed can help expedite the process if satellite dump or local satellite exists.

```
 [...]
#remove the satellite software media before making the resources available via the web
echo "-" >> /tmp/rc.local.out 2>&1
echo "- copy resource to web area" >> /tmp/rc.local.out 2>&1
echo "-" >> /tmp/rc.local.out 2>&1
date >> /tmp/rc.local.out 2>&1
/bin/rm /root/resources/redhat-rhn-satellite-5.3-server-x86_64-5-embedded-oracle.iso >>
/tmp/rc.local.out 2>&1
/bin/cp -r /root/resources /var/www/html/pub/  >> /tmp/rc.local.out 2>&1
 [...]
```

6. *subChannels.sh* - Subscribe to the required channels and set up nightly synchronizations

```
#!/bin/bash
satellite-sync --step=channels --channel=rhel-x86_64-server-5
satellite-sync --step=channels --channel=rhel-x86_64-server-vt-5
satellite-sync --step=channels --channel=rhel-x86_64-server-5-mrg-grid-1
satellite-sync --step=channels --channel=rhel-x86_64-server-5-mrg-management-1
satellite-sync --step=channels --channel=rhel-x86_64-server-cluster-5
```

```
satellite-sync --step=channels --channel=rhn-tools-rhel-x86_64-server-5
satellite-sync --step=channels --channel=rhel-x86_64-server-cluster-storage-5
satellite-sync --step=channels --channel=rhel-x86_64-rhev-mgmt-agent-5
satellite-sync --step=channels --channel=rhel-x86_64-server-5-mrg-grid-execute-1
satellite-sync --step=channels --channel=rhel-x86_64-server-5-mrg-messaging-1
echo "0 1 * * * perl -le 'sleep rand 9000' && satellite-sync --email >/dev/null 2>/dev/null" >>
/var/spool/cron/root
```

7. Start a background synchronization/download of the channels, attempt direct load first and then local satellite. While `satellite-sync` would be sufficient, possibly followed by another for verification, the steps listed can help expedite the process if satellite dump or local satellite exists.

```
 [...]
mkdir /pub > /tmp/satSync 2>&1

#either /pub should be mounted or the certificate must be retrieved
echo "irish.lab.bos.redhat.com:/pub /pub nfs defaults 0 0" >> /etc/fstab
/bin/mount /pub >> /tmp/satSync 2>&1
#wget http://irish.lab.bos.redhat.com/pub/RHN-ORG-TRUSTED-SSL-CERT -O /pub/RHN-ORG-
TRUSTED-SSL-CERT >> /tmp/satSync 2>&1 &

#These next two line attempt a direct import
satellite-sync --mount-point=/pub/projects/cloud/resources/wa/satDump -c rhel-x86_64-server-5 -c
rhel-x86_64-server-vt-5 -c rhel-x86_64-server-5-mrg-grid-1 -c rhel-x86_64-server-5-mrg-
management-1 -c rhel-x86_64-server-cluster-5 -c rhn-tools-rhel-x86_64-server-5 -c rhel-x86_64-
server-cluster-storage-5 -c rhel-x86_64-rhev-mgmt-agent-5 -c rhel-x86_64-server-5-mrg-grid-execute-
1 -c rhel-x86_64-server-5-mrg-messaging-1  >> /tmp/satSync 2>&1

( nice -n -15 satellite-sync --iss-parent=irish.lab.bos.redhat.com --ca-cert=/pub/RHN-ORG-TRUSTED-
SSL-CERT; nice -n -15 satellite-sync --iss-parent=irish.lab.bos.redhat.com --ca-cert=/pub/RHN-ORG-
TRUSTED-SSL-CERT; nice -n -15 satellite-sync ) >> /tmp/satSync 2>&1
 [...]
```

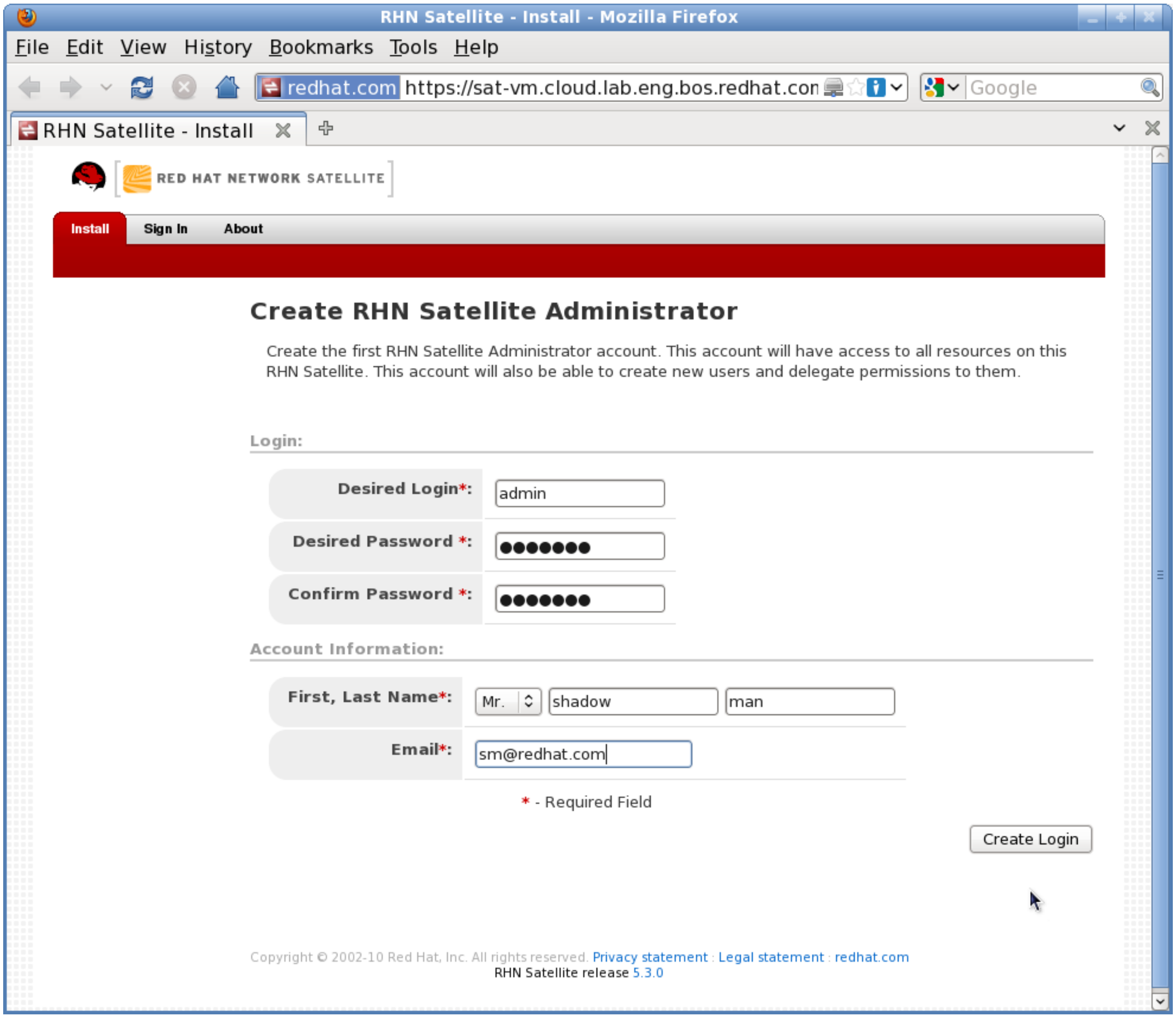
8. Copy actions to be performed on next boot of the satellite VM into a copy of the original */etc/rc.local* file.

```
 [...]
/bin/cat /etc/rc.d/rc.local.shipped  /root/resources/sat.rc.local2.add  >  /etc/rc.d/rc.local
```

6.2.4 Post Satellite Installation Actions

After the satellite has completed installation, the following actions must be performed:

1. User interaction is required to add the first user, by logging into the recently installed RHN satellite (e.g., *https://sat-vm.cloud.lab.eng.bos.redhat.com*). This can be performed anytime after the satellite software completes installation but must be performed prior to issuing the command to start the script in step 3 of this section.


RHN Satellite - Install - Mozilla Firefox
File Edit View History Bookmarks Tools Help
redhat.com https://sat-vm.cloud.lab.eng.bos.redhat.con
Google
RHN Satellite - Install
RED HAT NETWORK SATELLITE
Install
Sign In
About

Create RHN Satellite Administrator

Create the first RHN Satellite Administrator account. This account will have access to all resources on this RHN Satellite. This account will also be able to create new users and delegate permissions to them.

Login:

Desired Login*:	admin
Desired Password *:	●●●●●●●
Confirm Password *:	●●●●●●●

Account Information:

First, Last Name*:	Mr.	shadow	man
Email*:	sm@redhat.com		

* - Required Field

Copyright © 2002-10 Red Hat, Inc. All rights reserved. Privacy statement : Legal statement : redhat.com
RHN Satellite release 5.3.0

2. Before the next step, confirm all required channels have been synchronized. This output is logged in */var/log/rhn/rhn_server_satellite.log.* Issuing another `satellite-sync` takes only a few minutes if all channels are current, otherwise it should make the channels current.

```
satellite-sync
```

3. Call *post_satellite_build_up.sh*, recording the output.

```
/root/resources/post_satellite_build_up.sh 2>&1 | \
  tee /tmp/post_sat.out
```

The *post_satellite_build_up.sh* script:

- creates satellite organizations
- creates the organization default activation keys
- creates the activation keys for MRG systems
- provisions the first management system
- prepares satellite to install the JON and MRG Manager VMs
- prepares for the installation of the RHEV hosts
- imports the kickstart for use by tenant systems

```
#!/bin/bash

#Create the infrastructure and tenant organizations,
#Divide the entitlements, and define trusts.
echo "-"
echo "Creating Sat Orgs"
echo "-"
date
/root/resources/createOrgs.py

#Create default activation keys for the infrastructure and tenant organizations
echo "-"
echo "Creating default Activation Keys"
echo "-"
date
/root/resources/createDefActKeys.py

#Create the activation keys for the:
# - MRG Grid Manager in the infrastructure organization
# - MRG Grid Execute Node in the tenant organization
echo "-"
echo "Creating MRG Activation Key"
echo "-"
date
/root/resources/createMRGActKeys.py

#Build first management cluster member (mgmt1)
echo "-"
echo "Installing first management node"
echo "-"
date
/root/resources/instMgmt1.sh

#Add cobbler profile and system entries for the JON server and MRG VMs
echo "-"
echo "Prepping for Management VM creation"
echo "-"
date
/root/resources/prep_MgmtVMs.sh

#Prep and create RHEL/KVM hosts
echo "-"
echo "Prepping first RHEL host"
echo "-"
date
/root/resources/prep_RHELHost.sh

#Prep and create RHEV hosts
echo "-"
echo "Prepping first RHEV host"
echo "-"
date
/root/resources/prep_RHEVHost.sh

#Prep tenant kickstarts
echo "-"
echo "Import Tenant kickstarts"
echo "-"
date
/root/resources/prep_tenantKS.sh
```

a) *createOrgs.py* - Create organization, allocate entitlements, and create organizational trusts.

```
#!/usr/bin/python
"""
This script will create the infrastructure and tenant organizations,
divide the entitlements, and define trusts
"""

import xmlrpclib

SATELLITE_URL = "http://sat-vm.cloud.lab.eng.bos.redhat.com/rpc/api"
SATELLITE_LOGIN = "admin"
SATELLITE_PASSWORD = "24^gold"
INFRA_ORG_ENTITLEMENTS = 25

#open channel to satellite
client = xmlrpclib.Server(SATELLITE_URL, verbose=0)

#log in as administrator
key = client.auth.login(SATELLITE_LOGIN, SATELLITE_PASSWORD)

#create the infrastructure org
infraOrg = client.org.create(key, "infrastructure", "infra", "24^gold", "Mr.", "Shadow", "Man",
"sm@redhat.com", False)
```

```
#create the tenant org
tenantOrg = client.org.create(key, "tenant", "tenant", "24^gold", "Mr.", "Shadow", "Man",
"sm@redhat.com", False)

# retrieve all the system entitlements
SysEnts = client.org.listSystemEntitlements(key)

#loop through each entitlement
for SysE in SysEnts:
    #if INFRA_ORG_ENTITLEMENTS if more than 1/2 of the unallocated, only assign 1/2
    if SysE['unallocated'] > 2 * INFRA_ORG_ENTITLEMENTS:
        nEnt = INFRA_ORG_ENTITLEMENTS
    else:
        nEnt = SysE['unallocated'] / 2

    # assign infrastructure system entitlements printing an error on failure
    try:
        ret = client.org.setSystemEntitlements(key, infraOrg['id'], SysE['label'], nEnt)
    except:
        print "Unable to add: ",  nEnt, " entitlements of type: ", SysE['label'], " to org", infraOrg['id']

    # assign tenant system entitlements and print error on a failure
    try:
        ret = client.org.setSystemEntitlements(key, tenantOrg['id'], SysE['label'], SysE['unallocated'] -
nEnt)
    except:
        print "Unable to add: ", SysE['unallocated'] - nEnt, " entitlements of type: ", SysE['label'], " to
org", tenantOrg['id']

#Retrieve all the Software entitlements
SoftEnts = client.org.listSoftwareEntitlements(key)

for SoftE in SoftEnts:
    if SoftE['unallocated'] > 2 * INFRA_ORG_ENTITLEMENTS:
        nEnt = INFRA_ORG_ENTITLEMENTS
    else:
        nEnt = SoftE['unallocated'] / 2
    try:
        client.org.setSoftwareEntitlements(key, infraOrg['id'], SoftE['label'], nEnt)
    except:
        print "Unable to add: ", nEnt, " entitlements of type: ", SoftE['label'], " to org", infraOrg['id']
    try:
        client.org.setSoftwareEntitlements(key, tenantOrg['id'], SoftE['label'], SoftE['unallocated'] -
nEnt)
    except:
        print "Unable to add: ", SoftE['unallocated'] - nEnt, " entitlements of type: ", SoftE['label'], " to
org", tenantOrg['id']

#get all orgs
Orgs =  client.org.listOrgs(key)
```

```
#Double loop through the org setting trusts
o1 = 0
while o1 < len(Orgs) - 1:
    o2 = o1 + 1
    while o2 < len(Orgs):
        try:
            client.org.trusts.addTrust(key, Orgs[o1]['id'], Orgs[o2]['id'])
        except:
            print "Org:",  Orgs[o1]['id'], " failed to add trust for org: ",  Orgs[o2]['id']
        else:
            o2 += 1
    o1 += 1

client.auth.logout(key)
```

b) *createDefActKeys.py* – create universal default keys for each organizations

```
#!/usr/bin/python
"""
This script will create default Activation Keys for the infrastructure and tenant organizations
"""

import xmlrpclib

SATELLITE_URL = "http://sat-vm.cloud.lab.eng.bos.redhat.com/rpc/api"

INFRA_LOGIN = "infra"
INFRA_PASSWD = "24^gold"
INFRA_ENTITLE = [ 'monitoring_entitled', 'provisioning_entitled', 'virtualization_host_platform' ]
INFRA_PARENT = 'rhel-x86_64-server-5'
INFRA_CHILDREN = ['rhn-tools-rhel-x86_64-server-5', 'rhel-x86_64-server-vt-5', 'rhel-x86_64-server-cluster-5', \
                  'rhel-x86_64-server-cluster-storage-5', 'rhel-x86_64-server-5-mrg-grid-1', \
                  'rhel-x86_64-server-5-mrg-management-1', 'rhel-x86_64-server-5-mrg-messaging-1' ]
INFRA_PACKAGES = [ 'rhncfg', 'rhncfg-client', 'rhncfg-actions', 'osad', 'ntp' ]

TENANT_LOGIN = "tenant"
TENANT_PASSWD = "24^gold"
TENANT_ENTITLE = [ 'monitoring_entitled', 'provisioning_entitled' ]
TENANT_PARENT = 'rhel-x86_64-server-5'
TENANT_CHILDREN = ['rhn-tools-rhel-x86_64-server-5',  'rhel-x86_64-server-cluster-5', \
                  'rhel-x86_64-server-cluster-storage-5', \
                  'rhel-x86_64-server-5-mrg-grid-1', 'rhel-x86_64-server-5-mrg-messaging-1' ]
TENANT_PACKAGES = [ 'rhncfg', 'rhncfg-client', 'rhncfg-actions', 'osad', 'ntp' ]



"""
Create infrastructure key
"""
```

```
#open channel
client = xmlrpclib.Server(SATELLITE_URL, verbose=0)

#log into infrastructure org
key = client.auth.login(INFRA_LOGIN, INFRA_PASSWD)

#create key
infra_ak = client.activationkey.create(key, 'infraDefault', 'Default key for infrastructure org',
INFRA_PARENT, INFRA_ENTITLE, False)

#Add child channels
client.activationkey.addChildChannels(key, infra_ak, INFRA_CHILDREN)

#Add packages
client.activationkey.addPackageNames(key, infra_ak, INFRA_PACKAGES)

#log out from infrastructure channel
client.auth.logout(key)

"""
Create tenant key
"""
client = xmlrpclib.Server(SATELLITE_URL, verbose=0)

key = client.auth.login(TENANT_LOGIN, TENANT_PASSWD)

tenant_ak = client.activationkey.create(key, 'tenantDefault', 'Default key for tenant org',
TENANT_PARENT, TENANT_ENTITLE, True)

client.activationkey.addChildChannels(key, tenant_ak, TENANT_CHILDREN)

client.activationkey.addPackageNames(key, tenant_ak, TENANT_PACKAGES)

client.auth.logout(key)

print "Default Infrastructure activation key: ", infra_ak

print "Default Tenant activation key: ", tenant_ak
```

c) *createMRGActKeys.py* – create activation keys for MRG systems

```
#!/usr/bin/python
"""
This script will create the activation keys for the MRG Grid Manager (infrastructure org)
and MRG Grid Execute Node (tenant org)
"""

import xmlrpclib

SATELLITE_URL = "http://sat-vm.cloud.lab.eng.bos.redhat.com/rpc/api"
```

```
INFRA_LOGIN = "infra"
INFRA_PASSWD = "24^gold"
INFRA_ENTITLE = [ 'monitoring_entitled', 'provisioning_entitled' ]
INFRA_PARENT = 'rhel-x86_64-server-5'
INFRA_CHILDREN = ['rhn-tools-rhel-x86_64-server-5', 'rhel-x86_64-server-5-mrg-grid-1', \
                'rhel-x86_64-server-5-mrg-management-1', 'rhel-x86_64-server-5-mrg-messaging-1' ]
INFRA_PACKAGES = ['qpidd', 'sesame', 'qmf', 'condor', 'condor-qmf-plugins', 'cumin', \
                'perl-Frontier-RPC', 'rhncfg', 'rhncfg-client', 'rhncfg-actions', \
                'ntp', 'postgresql', 'postgresql-server' ]

TENANT_LOGIN = "tenant"
TENANT_PASSWD = "24^gold"
TENANT_ENTITLE = [ 'monitoring_entitled', 'provisioning_entitled' ]
TENANT_PARENT = 'rhel-x86_64-server-5'
TENANT_CHILDREN = ['rhn-tools-rhel-x86_64-server-5', 'rhel-x86_64-server-5-mrg-grid-1', \
                 'rhel-x86_64-server-5-mrg-messaging-1', 'rhel-x86_64-server-5-mrg-grid-execute-1' ]
TENANT_PACKAGES = ['qpidd', 'sesame', 'qmf', 'condor', 'condor-qmf-plugins', 'cumin', \
                'perl-Frontier-RPC', 'rhncfg', 'rhncfg-client', 'rhncfg-actions', \
                'ntp', 'postgresql', 'postgresql-server' ]

"""
Create Key for MRG Grid Manager
"""

#open channel
client = xmlrpclib.Server(SATELLITE_URL, verbose=0)

#log into infrastructure org
key = client.auth.login(INFRA_LOGIN, INFRA_PASSWD)

#create key
infra_ak = client.activationkey.create(key, 'infraMRGGrid', 'Key for MRG Manager',
INFRA_PARENT, INFRA_ENTITLE, False)

#Add child channels
client.activationkey.addChildChannels(key, infra_ak, INFRA_CHILDREN)

#Add packages
client.activationkey.addPackageNames(key, infra_ak, INFRA_PACKAGES)

#log out from infrastructure channel
client.auth.logout(key)

"""
Create MRG Grid Exec Node Activation Key
"""

client = xmlrpclib.Server(SATELLITE_URL, verbose=0)

key = client.auth.login(TENANT_LOGIN, TENANT_PASSWD)
```

```
tenant_ak = client.activationkey.create(key, 'tenantMRGGridExec', 'Key for MRG Grd Exec
Nodes', TENANT_PARENT, TENANT_ENTITLE, False)

client.activationkey.addChildChannels(key, tenant_ak, TENANT_CHILDREN)

client.activationkey.addPackageNames(key, tenant_ak, TENANT_PACKAGES)

client.auth.logout(key)


#print out the defined keys
print "MRG Manager activation key: ", infra_ak

print "MRG Grid Exec activation key: ", tenant_ak
```

d) *instMgmt1.sh* – start the provisioning of the first management system (the details of which are in the next section) that:

- configures cobbler to boot system
- moves network to first in boot order
- powers on/ reset the system
- moves network to last in boot order

```
#!/bin/bash
# source env vars
echo "-"
echo "Sourcing variables"
echo "-"
date
if [[ -x varDefs.sh ]] ; then
  source varDefs.sh
elif [[ -x /root/varDefs.sh ]] ; then
  source /root/varDefs.sh
elif [[ -x /root/resources/varDefs.sh ]] ; then
  source /root/resources/varDefs.sh
elif [[ -x /root/distro/resources/varDefs.sh ]] ; then
  source /root/distro/resources/varDefs.sh
else
  echo "didn't find a varDefs.sh file"
fi

#prepare cobbler to boot this node
echo "-"
echo "Adding cobbler system entry for Mgmt1"
echo "-"
date
cobbler profile add --name=$MGMT_PROFILE --distro=$MGMT_DISTRO
--kickstart=/root/resources/$MGMT_KS
cobbler system add --name=$MGMT1_NAME --profile=$MGMT_PROFILE
--mac=$MGMT1_MAC --ip=$MGMT1_IP --hostname=MGMT1_NAME
```

```
--ksmeta="NAME=$MGMT1_NAME NETIP=$MGMT1_IP ICIP=$MGMT1_ICIP
ICNAME=$MGMT1_IC" –kopts="console=ttyS0,115200 nostorage"
cobbler sync

# set to boot PXE
echo "-"
echo "Setting Mgmt to boot PXE"
echo "-"
date
while [[ ! `ilocommand -i //${LOGIN}:${MGMT1_ILO_PW}@${MGMT1_ILO} set
/system1/bootconfig1/bootsource5 bootorder=1 | grep status=0` ]]; do sleep 2; done

#reset blade, power up also in case it was powered off
echo "-"
echo "Booting Mgmt1"
echo "-"
date
ilocommand -i //${LOGIN}:${MGMT1_ILO_PW}@${MGMT1_ILO} power reset
ilocommand -i //${LOGIN}:${MGMT1_ILO_PW}@${MGMT1_ILO} power on

#wait a reasonable amount of time and change the boot order to make network last
echo "-"
echo "Waiting then resetting boot order "
echo "-"
date
sleep 240
while [[ ! `ilocommand -i //${LOGIN}:${MGMT1_ILO_PW}@${MGMT1_ILO} set
/system1/bootconfig1/bootsource5 bootorder=5 | grep status=0` ]]; do sleep 2; done
```

e) *prep_MgmtVMs.sh* – create profile for RHEL Management VMs

```
#!/bin/bash

# source env vars
if [[ -x varDefs.sh ]] ; then
  source varDefs.sh
elif [[ -x /root/varDefs.sh ]] ; then
  source /root/varDefs.sh
elif [[ -x /root/resources/varDefs.sh ]] ; then
  source /root/resources/varDefs.sh
elif [[ -x /root/distro/resources/varDefs.sh ]] ; then
  source /root/distro/resources/varDefs.sh
else
  echo "didn't find a varDefs.sh file"
fi

#Add cobbler profile and system entries for the JON server VM
cobbler profile add --name=${JON_PROFILE} --distro=${MGMT_DISTRO}
--kickstart=/root/resources/${JON_KS}
cobbler system add --name=${JON_NAME} --profile=${JON_PROFILE} --mac=${JON_MAC}
--ip=${JON_IP} --hostname=${JON_NAME}
```

```
#Add cobbler profile and system entries for the MRG VM
cobbler profile add --name=${MRGGRID_PROFILE} --distro=${MGMT_DISTRO}
--kickstart=/root/resources/${MRGGRID_KS}
cobbler system add --name=${MRGGRID_NAME} --profile=${MRGGRID_PROFILE} --mac=$
{MRGGRID_MAC} --ip=${MRGGRID_IP} --hostname=${MRGGRID_NAME}

#Rebuild the contents of /tftpboot
cobbler sync
```

f) *prep_RHELHost.sh* – prepare kickstart and activation keys for RHEL/KVM hosts

```
#!/bin/bash

# This script will prepare the infrastructure to add RHEL/KVM host to RHEV

# Create the appropriate activation key
/root/resources/createRHELHKey.py

# Import the Kickstart into satellite
/root/resources/importKS_infra.py /root/resources/rhelHost.ks

# Associate the key with the kickstart
/root/resources/assocKeyKS_infra.py RHELH rhelHost
```

i) *createRHELHKey.py* – Create activation key for RHEL/KVM hosts

```
#!/usr/bin/python
"""
This script will create the activation key for RHEL/KVM system to be used for RHEV Hosts
"""

import xmlrpclib

SATELLITE_URL = "http://sat-vm.cloud.lab.eng.bos.redhat.com/rpc/api"

INFRA_LOGIN = "infra"
INFRA_PASSWD = "24^gold"
INFRA_ENTITLE = """monitoring_entitled
provisioning_entitled
virtualization_host_platform
"""
INFRA_PARENT = 'rhel-x86_64-server-5'
INFRA_CHILDREN = """rhn-tools-rhel-x86_64-server-5
rhel-x86_64-server-vt-5
rhel-x86_64-rhev-mgmt-agent-5
"""
INFRA_PACKAGES = """rhn-virtualization-host
osad
"""

"""
Create Activation Key for RHEL-H
```

```
"""

#open channel
client = xmlrpclib.Server(SATELLITE_URL, verbose=0)

#log into infrastructure org
key = client.auth.login(INFRA_LOGIN, INFRA_PASSWD)

#create key
infra_ak = client.activationkey.create(key, 'RHELH', 'Key for RHEL/KVM based RHEL
Hosts', INFRA_PARENT, INFRA_ENTITLE.splitlines(), False)

#Add child channels
client.activationkey.addChildChannels(key, infra_ak, INFRA_CHILDREN.splitlines())

#Add packages
client.activationkey.addPackageNames(key, infra_ak, INFRA_PACKAGES.splitlines())

#log out from infrastructure channel
client.auth.logout(key)
```

ii) *importKS_infra.py* – import supplied kickstart into the infrastructure organization

```
#!/usr/bin/python
"""
This script will attempt to create a kickstart form the content of the file passed.
The ks name will be derived from the filename - if basename less than 6 chars date is appended
the org, kstree and VIRT type are determined by the global variable below
"""

import xmlrpclib
import os.path
import sys
import time
import datetime

SATELLITE_URL = "http://sat-vm.cloud.lab.eng.bos.redhat.com/rpc/api"

INFRA_LOGIN = "infra"
INFRA_PASSWD = "24^gold"
KSTREE = 'ks-rhel-x86_64-server-5-u5'
VIRT = "none"

def main():
    if len(sys.argv) < 2:
        print "Usage: ",sys.argv[0]," kickstart"
        sys.exit(-2);

    # retreieve the passed parameter
    fileName = sys.argv[1]
    filePtr = open(fileName, 'r')

    ksName =  os.path.basename(fileName).split('.')[0]
    if len(ksName) < 6:
        today = datetime.date.today()
        ksName =  ksName + '_' + today.strftime("%m%d")

    #open channel
    client = xmlrpclib.Server(SATELLITE_URL, verbose=0)

    #log into infrastructure org
    key = client.auth.login(INFRA_LOGIN, INFRA_PASSWD)

    client.kickstart.importFile(key, ksName, VIRT, KSTREE, filePtr.read())

    #log out from infrastructure channel
    client.auth.logout(key)

if __name__ == "__main__":
    sys.exit(main())
```

iii) *assocKeyKS_infra.py* – Associate an activation key with an existing kickstart

```
#!/usr/bin/python
"""
This script will attempt to associate the passed activation key to the kickstart
  related to the label that was passed. the org is determined by the login global variable below
"""

import xmlrpclib
import sys


SATELLITE_URL = "http://sat-vm.cloud.lab.eng.bos.redhat.com/rpc/api"


INFRA_LOGIN = "infra"
INFRA_PASSWD = "24^gold"



def main():
    if len(sys.argv) < 3:
        print "Usage: ",sys.argv[0]," activationKey kickstart"
        sys.exit(-2);

    # retreieve the passed parameter
    AKName = sys.argv[1]

    KSName = sys.argv[2].split(':')[0]
    #open channel
    client = xmlrpclib.Server(SATELLITE_URL, verbose=0)

    #log into infrastructure org
    key = client.auth.login(INFRA_LOGIN, INFRA_PASSWD)


    aKeys = client.activationkey.listActivationKeys(key)
    for iKey in aKeys:
        if iKey['key'] == AKName or iKey['key'].split('-')[1] == AKName:
            fKey = iKey
            break
    else:
        print "No activation key matched: ", AKName
        client.auth.logout(key)
        exit(-3)

    aKS = client.kickstart.listKickstarts(key)
    for iKS in aKS:
        if iKS['label'] == KSName:
            fKS = iKS
            break
    else:
        print "No kickstart matched: ", KSName
        client.auth.logout(key)
        exit(-4)

    client.kickstart.profile.keys.addActivationKey(key, fKS['label'], fKey['key'])

    #log out from infrastructure channel
    client.auth.logout(key)

if __name__ == "__main__":
    sys.exit(main())
```

g) *prep_RHEVHost.sh* – script to prepare the required storage that:

- installs the RHEV hypervisor RPM
- generates the PXE boot file from installed hypervisor
- creates the cobbler distribution and profile

```
#!/bin/bash

#source variables
if [[ -x varDefs.sh ]]
then
  source varDefs.sh
elif [[ -x /root/varDefs.sh ]]
then
  source /root/varDefs.sh
elif [[ -x /root/resources/varDefs.sh ]]
```

```
then
  source /root/resources/varDefs.sh
elif [[ -x /root/distro/resources/varDefs.sh ]]
then
  source /root/distro/resources/varDefs.sh
else
  echo "didn't find a varDefs.sh file"
fi

# Install RHEV-H
# find the rpm file names
RPMFILE=`ls /root/resources/rhev-hypervisor-*.noarch.rpm`

# calculate the number of RPMs for each tool
RPMNUM=`echo $RPMFILE | wc -w`

if [[ ${RPMNUM} -eq 0 ]]
then
  echo "No rhev hypervisor RPM file!"
elif [[ ${RPMNUM} -gt 1 ]]
then
 echo "More than one hypervisor RPM file!"
else
  yum -y --nogpgcheck localinstall ${RPMFILE}
fi

# Generate files required for PXE boot
echo -e "-\nGenerating files required for PXE boot ...\n-"
cd /usr/share/rhev-hypervisor

# Set root passwd
passwd='$1$XpieUBDF$NTX/4tcT/A5P7iZ.voQl01'

# temporary resolution to issues (bnx2x panic dumps, cciss disks, etc.)
#/bin/mv rhev-hypervisor.iso rhev-hypervisor.iso.hold
#/bin/cp /root/resources/rhev-hypervisor-5.5-2.2.5cciss.el5_5rhev2_2.iso rhev-hypervisor.iso

livecd-iso-to-pxeboot /usr/share/rhev-hypervisor/rhev-hypervisor.iso

# Create cobbler distribution entry
echo -e "-\nCreating cobbler distribution entry ...\n-"
cobbler distro add --name=${RHEVH_PROFILE} --kernel=/usr/share/rhev-
hypervisor/tftpboot/vmlinuz0 --initrd=/usr/share/rhev-hypervisor/tftpboot/initrd0.img
--kopts="rootflags=loop root=/rhev-hypervisor.iso rootfstype=auto liveimg rhn_url=https://$
{SAT_IP}/ rhn_activationkey=2-RHELH"

# Create cobbler profile entry to use the distro
echo -e "-\nCreating cobbler profile entry ...\n-"
cobbler profile add --name=${RHEVH_PROFILE} --distro=${RHEVH_PROFILE}
--kopts="storage_init=/dev/cciss/c0d0 storage_vol=:::::: BOOTIF=eth0 management_server=$
```

```
{RHEVM_IP} netconsole=${RHEVM_IP} rootpw=${passwd} ssh_pwauth=1 local_boot"

# Update cobbler system files
cobbler sync
```

h) *prep_tenantKS.sh* performs the following:

- creates custom channel on satellite for applications
- creates an activation key for the VM running the Java application
- imports these kickstart files
    - basic RHEL55
    - RHEL55 grid execute node
    - RHEL55 gird execute node with render application
    - RHEL55 with java application
    - RHEL55 with JBoss Seam booking demo application
- associates activation keys to selected VM (others use the default)
- associates the application channel GPG key with the Java kickstart

```
#!/bin/bash

# This script will preload the tenant organization with kickstarts for tenant deployment

# create apps channel on the satellite
/root/resources/prep_AppsChannel.sh

# create any activation keys
/root/resources/createJavaActKey.py

# import kickstarts
/root/resources/importKS_tenant.py /root/resources/rhel55_basic.ks
/root/resources/importKS_tenant.py /root/resources/rhel55_mrg_exec.ks
/root/resources/importKS_tenant.py /root/resources/rhel55_mrg_render.ks
/root/resources/assocKeyKS_tenant.py tenantMRGGridExec rhel55_mrg_exec
/root/resources/assocKeyKS_tenant.py tenantMRGGridExec rhel55_mrg_render
/root/resources/importKS_tenant.py /root/resources/rhel55_java.ks
/root/resources/assocKeyKS_tenant.py tenantrheljava rhel55_java
/root/resources/importKS_tenant.py /root/resources/rhel55_jboss.ks

# associate GPG keys with kickstarts
/root/resources/addGPGKey_tenant.py rhel55_java /var/www/html/pub/APP-RPM-GPG-KEY
```

i) *prep_AppsChannel.sh* does the following:

- generates GPG key and signs the javaApp RPM
- creates application custom channel
- pushes the RPM to the created channel

```
#!/bin/bash

# source env vars
```

```
if [[ -x varDefs.sh ]] ; then
  source varDefs.sh
elif [[ -x /root/varDefs.sh ]] ; then
  source /root/varDefs.sh
elif [[ -x /root/resources/varDefs.sh ]] ; then
  source /root/resources/varDefs.sh
elif [[ -x /root/distro/resources/varDefs.sh ]] ; then
  source /root/distro/resources/varDefs.sh
else
  echo "didn't find a varDefs.sh file"
fi

#Generate a key and sign the package
/root/resources/AppPGPKey.sh

# Create our Application channel
/root/resources/createAppSatChannel.py

# put rpm file locally on satellite
mkdir -p /var/satellite/mychannels/ourapps

#Figure out the package name
JFILE=`ls /root/resources/javaApp-*.noarch.rpm`
JNUM=`echo $JFILE | wc -w`
if [[ $JNUM -eq 0 ]]
then
  echo "No javaApp rpm!"
  exit -2
elif [[ $JNUM -gt 1 ]]
then
 echo "Too many javaApp RPMs!"
 exit -3
fi

# copy the file to a known area on the satellite
/bin/cp ${JFILE} /var/satellite/mychannels/ourapps

# Push the rpm into satellite
rhnpush -v -courapps  --server=http://localhost/APP
--dir=/var/satellite/mychannels/ourapps -u ${SAT_TENANT_USER} -p $
{SAT_TENANT_PW}
```

- *AppPGPKey.sh* does the following:
  - generates GPG key
  - prepares ~/.rpmmacros for signing packages
  - signs the RPM
  - makes the public key available

```
#!/bin/bash
```

```
# This script will generate a GPG key to use for the App Channel,
# resign the javaApp package, and make the pub key available.

#  -- create profile with GPG key?

# Generate the key
gpg --batch --gen-key /root/resources/AppPGP

# Determine the key name
KNAME=`gpg --list-keys --fingerprint --with-colons |grep Vijay | cut -f5 -d':' | cut -c9-`

# Prep for signing
cat <<EOF>>~/.rpmmacros
%_signature gpg
%_gpg_name ${KNAME}
EOF

# Determine the package name
JFILE=`ls /root/resources/javaApp-*.noarch.rpm`
JNUM=`echo $JFILE | wc -w`
if [[ $JNUM -eq 0 ]]
then
  echo "No javaApp rpm!"
  exit -2
elif [[ $JNUM -gt 1 ]]
then
 echo "More than one javaApp RPMs!"
 exit -3
fi

# Resign the key
/root/resources/sign_rpm.py ${JFILE} "Cloud Foundations"

# Export the public key
gpg --export -a 'Vijay Trehan' > javaApp_pubKey.txt

# Copy the key to satellite web public area
/bin/cp javaApp_pubKey.txt /var/www/html/pub/APP-RPM-GPG-KEY
```

- *sign_rpm.py* – python expect script to sign the package

```
#!/usr/bin/python
import sys, os, pexpect

filename = sys.argv[1]
passphrase = sys.argv[2]

signproc = pexpect.spawn("rpm --resign %s" % filename)
match = signproc.expect(["Enter pass phrase:"])

signproc.sendline(passphrase)
```

```
match = signproc.expect([pexpect.EOF])
```

- *createAppSatChannel.py* – create Application custom channel in Satellite

```
#!/usr/bin/python
""""
This script will create a custom channel for custom applications
""""

import xmlrpclib

SATELLITE_URL = "http://sat-vm.cloud.lab.eng.bos.redhat.com/rpc/api"

TENANT_LOGIN = "tenant"
TENANT_PASSWD = "24^gold"

NAME = 'ourapps'
LABEL = 'ourapps'
SUMMARY = 'Custom Applications'
ARCH = 'channel-x86_64'
PARENT = 'rhel-x86_64-server-5'

""""
Create channel for Application
""""

#open channel
client = xmlrpclib.Server(SATELLITE_URL, verbose=0)

#log into infrastructure org
key = client.auth.login(TENANT_LOGIN, TENANT_PASSWD)

#create key
try:
    client.channel.software.create(key, LABEL, NAME, SUMMARY, ARCH, PARENT)
except:
    print "Failed to create OurApps Channel"

#log out from infrastructure channel
client.auth.logout(key)
```

ii) *createJavaActkey.py* – create activation key used for RHEL guest running the javaApp

```
#!/usr/bin/python
""""
This script will create the activation key for the RHEL Java Application (tenant org)
""""

import xmlrpclib
```

```
SATELLITE_URL = "http://sat-vm.cloud.lab.eng.bos.redhat.com/rpc/api"

TENANT_LOGIN = "tenant"
TENANT_PASSWD = "24^gold"
TENANT_ENTITLE = [ 'monitoring_entitled', 'provisioning_entitled' ]
TENANT_PARENT = 'rhel-x86_64-server-5'
TENANT_CHILDREN = ['rhn-tools-rhel-x86_64-server-5', 'ourapps' ]
TENANT_PACKAGES = [ 'ntp', 'javaApp' ]

"""
Create Key for Java Application
"""

#open channel
client = xmlrpclib.Server(SATELLITE_URL, verbose=0)

#log into tenant org
key = client.auth.login(TENANT_LOGIN, TENANT_PASSWD)

#create key
tenant_ak = client.activationkey.create(key, 'tenantrheljava', 'Key for RHEL Node with
javaApp', TENANT_PARENT, TENANT_ENTITLE, False)

#Add child channels
client.activationkey.addChildChannels(key, tenant_ak, TENANT_CHILDREN)

#Add packages
client.activationkey.addPackageNames(key, tenant_ak, TENANT_PACKAGES)

#log out from infrastructure channel
client.auth.logout(key)


#print out the defined key
print "JavaApp activation key: ", tenant_ak
```

iii) *importKS_tenant.py* – import kickstart into RHN satellite

```
#!/usr/bin/python
"""
  This script will create a kickstart from the content of the file passed.
  The ks name will be derived from the filename - if basename is less than 6 chars, date is
appended
  The org, kstree and VIRT type are determined by the global variable below
"""

import xmlrpclib
import os.path
import sys
import time
import datetime
```

```
SATELLITE_URL = "http://sat-vm.cloud.lab.eng.bos.redhat.com/rpc/api"

INFRA_LOGIN = "tenant"
INFRA_PASSWD = "24^gold"
KSTREE = 'ks-rhel-x86_64-server-5-u5'
VIRT = "none"

def main():
    if len(sys.argv) < 2:
        print "Usage: ",sys.argv[0]," kickstart"
        sys.exit(-2);

    # retreieve the passed parameter
    fileName = sys.argv[1]

    filePtr = open(fileName, 'r')

    ksName =  os.path.basename(fileName).split('.')[0]
    if len(ksName) < 6:
        today = datetime.date.today()
        ksName =  ksName + '_' + today.strftime("%m%d")

    #open channel
    client = xmlrpclib.Server(SATELLITE_URL, verbose=0)

    #log into infrastructure org
    key = client.auth.login(INFRA_LOGIN, INFRA_PASSWD)

    client.kickstart.importFile(key, ksName, VIRT, KSTREE, filePtr.read())

    #log out from infrastructure channel
    client.auth.logout(key)

if __name__ == "__main__":
    sys.exit(main())
```

iv) *assocKeyKS_tenant.py* – associates an existing activation key with an existing kickstart for the tenant organization

```
#!/usr/bin/python
"""
This script will attempt to associate the passed activation key to the kickstart
  related to the label that was passed. the org is determined by the login global variable below
"""

import xmlrpclib
import sys


SATELLITE_URL = "http://sat-vm.cloud.lab.eng.bos.redhat.com/rpc/api"
```

```
TENANT_LOGIN = "tenant"
TENANT_PASSWD = "24^gold"



def main():
    if len(sys.argv) < 3:
        print "Usage: ",sys.argv[0]," activationKey kickstart"
        sys.exit(-2);

    # retreieve the passed parameter
    AKName = sys.argv[1]

    KSName = sys.argv[2].split(':')[0]

    #open channel
    client = xmlrpclib.Server(SATELLITE_URL, verbose=0)

    #log into infrastructure org
    key = client.auth.login(TENANT_LOGIN, TENANT_PASSWD)


    aKeys = client.activationkey.listActivationKeys(key)
    for iKey in aKeys:
        if iKey['key'] == AKName or iKey['key'].split('-')[1] == AKName:
            fKey = iKey
            break
    else:
        print "No activation key matched: ", AKName
        client.auth.logout(key)
        exit(-3)


    aKS = client.kickstart.listKickstarts(key)
    for iKS in aKS:
        if iKS['label'] == KSName:
            fKS = iKS
            break
    else:
        print "No kickstart matched: ", KSName
        client.auth.logout(key)
        exit(-4)

    client.kickstart.profile.keys.addActivationKey(key, fKS['label'], fKey['key'])

    #log out from infrastructure channel
    client.auth.logout(key)
```

```
if __name__ == "__main__":
    sys.exit(main())
```

v) *addGPGKey_tenant.py* - loads the GPG key into satellite and associates it with a stated kickstart

```
#!/usr/bin/python
"""
This script will attempt to add a GPG key to an existing kickstart
"""

import xmlrpclib
import os.path
import sys
import time
import datetime

SATELLITE_URL = "http://sat-vm.cloud.lab.eng.bos.redhat.com/rpc/api"

INFRA_LOGIN = "tenant"
INFRA_PASSWD = "24^gold"

def main():
    if len(sys.argv) < 3:
        print "Usage: ",sys.argv[0]," kickstart GPGKEY_pub.txt"
        sys.exit(-2);

    # retreieve the passed parameter
    kickstart = sys.argv[1]
    fileName = sys.argv[2]

    filePtr = open(fileName, 'r')
    kDesc = os.path.basename(fileName).split('.')[0]

    #open channel
    client = xmlrpclib.Server(SATELLITE_URL, verbose=0)

    #log into infrastructure org
    key = client.auth.login(INFRA_LOGIN, INFRA_PASSWD)

    # create key
    client.kickstart.keys.create(key, kDesc, 'GPG', filePtr.read())

    # add key to kicktart
    client.kickstart.profile.system.addKeys(key, kickstart, [ kDesc ])

    #log out from infrastructure channel
    client.auth.logout(key)

if __name__ == "__main__":
    sys.exit(main())
```

6.3 Provision First Management Node

The invocation of *post_satellite_build_up.sh*, among other actions, starts the installation of the first management node. The same kickstart file is used for both management nodes using variables defining hard-coded network parameters (NAME, NETIP, ICIP, ICNAME) to differentiate between the systems. As the first management node boots, it becomes the host to the satellite VM, create the other management VMs, coordinate with the other management node and form the cluster.

Kickstart Install

The multi-use kickstart installs the system with the software needed for clustering and KVM virtualization. Two hardcoded networks are configured, one for public access and the other for cluster interconnect traffic. Additionally, the post script:

- retrieves the time from a trusted source and sets the platform hardware clock
- configures the firewall to work with clustering, virtualization, NFS and other services
- creates a minimal host file
- updates LVM use of preferred names and types
- downloads required files from the satellite public area
- converts the public network to a bridge to be used for virtualization
- registers the system with the RHN satellite server
- prepares the system to be a named slave server
- configures NFS for use with a firewall
- disables LRO
- updates all software
- prepares actions to be performed on the next boot

mgmt1.ks

```
install
text
key <Installation Number>
#lang en_US
lang en_US.UTF-8
keyboard us
skipx
url --url http://sat-vm.cloud.lab.eng.bos.redhat.com/ks/dist/ks-rhel-x86_64-server-5-u5
network --device eth0 --bootproto static --ip $NETIP --netmask 255.255.248.0 --gateway 10.16.143.254
--nameserver 10.16.136.1,10.16.255.2 --hostname $NAME
network --device eth1 --bootproto static --ip $ICIP --netmask 255.255.255.0 --hostname $ICNAME
network --device eth2 --onboot no --bootproto dhcp --hostname mgmt1.cloud.lab.eng.bos.redhat.com
network --device eth3 --onboot no --bootproto dhcp --hostname mgmt1.cloud.lab.eng.bos.redhat.com
network --device eth4 --onboot no --bootproto dhcp --hostname mgmt1.cloud.lab.eng.bos.redhat.com
network --device eth5 --onboot no --bootproto dhcp --hostname mgmt1.cloud.lab.eng.bos.redhat.com
network --device eth6 --onboot no --bootproto dhcp --hostname mgmt1.cloud.lab.eng.bos.redhat.com
network --device eth7 --onboot no --bootproto dhcp --hostname mgmt1.cloud.lab.eng.bos.redhat.com
services --enabled=multipathd,ntpd
```

```
device scsi cciss
zerombr
clearpart --all --initlabel
part /boot --fstype=ext3 --size=200
part pv.01 --size=1000 --grow
part swap --size=10000   --maxsize=20000
volgroup mgmtNode1VG pv.01
logvol / --vgname=mgmtNode1VG --name=rootvol --size=1000 --grow
bootloader --location mbr
timezone America/New_York
auth --enablemd5 --enableshadow
rootpw --iscrypted $1$bdeGv4w8$zkMITYuaYvSPh2.W.nD.D.
selinux --permissive
reboot
firewall --enabled
repo --name=Cluster --baseurl=http://sat-vm.cloud.lab.eng.bos.redhat.com/ks/dist/ks-rhel-x86_64-server-5-u5/Cluster
repo --name=ClusterStorage --baseurl=http://sat-vm.cloud.lab.eng.bos.redhat.com/ks/dist/ks-rhel-x86_64-server-5-u5/ClusterStorage
repo --name=VT --baseurl=http://sat-vm.cloud.lab.eng.bos.redhat.com/ks/dist/ks-rhel-x86_64-server-5-u5/VT

%packages

@ Base
@admin-tools
@cluster-storage
@clustering
@core
@development-libs
@development-tools
@dialup
@editors
@gnome-desktop
@gnome-software-development
@graphical-internet
@graphics
@java
@kvm
@legacy-software-support
@office
@printing
@sound-and-video
@text-internet
@web-server
@x-software-development
@base-x
bridge-utils
device-mapper-multipath
emacs
```

```
fipscheck
imake
kexec-tools
libsane-hpaio
mesa-libGLU-devel
ntp
perl-XML-SAX
perl-XML-NamespaceSupport
python-imaging
python-dmidecode
pexpect
sg3_utils
sgpio
xorg-x11-utils
xorg-x11-server-Xnest
xorg-x11-server-Xvfb
yum-utils
bind

%post
(
# set system time from server then set time to the hwclock
/usr/sbin/ntpdate -bu 10.16.255.2
hwclock --systohc
cat <<EOF>/etc/ntp.conf
restrict default kod nomodify notrap nopeer noquery
restrict 10.16.136.0 mask 255.255.248.0
restrict 127.0.0.1
server 10.16.255.2
server 10.16.255.3
driftfile /var/lib/ntp/drift
keys /etc/ntp/keys
EOF

#update the hostname, BZ-593093
/bin/sed -ic "/HOSTNAME=/D" /etc/sysconfig/network
echo "HOSTNAME=$NAME" >> /etc/sysconfig/network

#configure the iptables
cat <<'EOF'> /etc/sysconfig/iptables
*nat
:PREROUTING ACCEPT [32:4811]
:POSTROUTING ACCEPT [328:24055]
:OUTPUT ACCEPT [314:20588]
-A POSTROUTING -s 192.168.122.0/255.255.255.0 -d ! 192.168.122.0/255.255.255.0 -j MASQUERADE
COMMIT
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [23214:3084596]
```

```
:RH-Firewall-1-INPUT - [0:0]
-A INPUT -i virbr0 -p udp -m udp --dport 53 -j ACCEPT
-A INPUT -i virbr0 -p tcp -m tcp --dport 53 -j ACCEPT
-A INPUT -i virbr0 -p udp -m udp --dport 67 -j ACCEPT
-A INPUT -i virbr0 -p tcp -m tcp --dport 67 -j ACCEPT
-A INPUT -j RH-Firewall-1-INPUT
-A FORWARD -d 192.168.122.0/255.255.255.0 -o virbr0 -m state --state RELATED,ESTABLISHED -j
ACCEPT
-A FORWARD -s 192.168.122.0/255.255.255.0 -i virbr0 -j ACCEPT
-A FORWARD -i virbr0 -o virbr0 -j ACCEPT
-A FORWARD -o virbr0 -j REJECT --reject-with icmp-port-unreachable
-A FORWARD -i virbr0 -j REJECT --reject-with icmp-port-unreachable
-A FORWARD -j RH-Firewall-1-INPUT
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 53 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 53 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 875 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 875 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 892 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 892 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 32769 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 32769 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 32803 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 32803 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 662 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 662 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 111 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 111 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 2049 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 2049 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 2020 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 2020 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 49152:49216 -j ACCEPT
-A RH-Firewall-1-INPUT -m physdev  --physdev-is-bridged -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m state --state NEW -m multiport --dports 50007 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m state --state NEW -m multiport --dports 50006,50007,50008,50009 -j
ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m state --state NEW -m multiport --dports 21064 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m state --state NEW -m multiport --dports 14567 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m state --state NEW -m multiport --dports 11111 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m state --state NEW -m multiport --dports 41966,41967,41968,41969 -j
ACCEPT
-A RH-Firewall-1-INPUT -p udp -m state --state NEW -m multiport --dports 5404,5405 -j ACCEPT
-A RH-Firewall-1-INPUT -i lo -j ACCEPT
-A RH-Firewall-1-INPUT -p icmp -m icmp --icmp-type any -j ACCEPT
-A RH-Firewall-1-INPUT -p esp -j ACCEPT
-A RH-Firewall-1-INPUT -p ah -j ACCEPT
-A RH-Firewall-1-INPUT -d 224.0.0.251 -p udp -m udp --dport 5353 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 123 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 631 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 631 -j ACCEPT
```

```
-A RH-Firewall-1-INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m state --state NEW -m tcp --dport 443 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT
-A RH-Firewall-1-INPUT -j REJECT --reject-with icmp-host-prohibited
COMMIT
EOF

#create host file with cluster members
cat <<'EOF'> /etc/hosts
127.0.0.1               localhost.localdomain localhost
::1                     localhost6.localdomain6 localhost6
10.16.136.10            mgmt1.cloud.lab.eng.bos.redhat.com mgmt1
10.16.136.15            mgmt2.cloud.lab.eng.bos.redhat.com mgmt2
192.168.136.10          mgmt1-ic.cloud.lab.eng.bos.redhat.com mgmt1-ic
192.168.136.15          mgmt2-ic.cloud.lab.eng.bos.redhat.com mgmt2-ic
EOF

# updated LVM's preferred names and types
sed -i -e 's/    preferred_names = \[ \]/preferred_names = \[ "^\/dev\/mapper\/" \]/' -e 's/# types = \[ "fd",
16 \]/types = \[ "device-mapper", 16 \]/' /etc/lvm/lvm.conf

#get required files
cd /root
wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/ilolib-current.noarch.rpm
wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/oalib-current.noarch.rpm
wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/salib-current.noarch.rpm
wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/vcmlib-current.noarch.rpm
wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/instpyCmds.sh
wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/prep_stor_mgmt.sh
wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/map_fc_aliases.sh
wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/multipath.conf.template
wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/buildMpathAliases.sh
wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/mgmt.rc.local.add
wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/cvt_br.sh
wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/cluster.conf
wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/createMgmtVMs.sh
wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/named.conf.slave -O /etc/named.conf
wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/varDefs.sh
chmod +x *.sh

#convert eth0 to a bridge named cloud0
./cvt_br.sh eth0 cloud0

#register system with satellite
rpm -ivh http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/rhn-org-trusted-ssl-cert-1.0-1.noarch.rpm
rhnreg_ks --profilename=$NAME --activationkey=2-infraDefault --serverUrl=https://sat-
vm.cloud.lab.eng.bos.redhat.com/XMLRPC --sslCACert=/usr/share/rhn/RHN-ORG-TRUSTED-SSL-CERT

#start named (as a slave)
```

```
setsebool -P named_write_master_zones on
chmod 770 /var/named
chkconfig named on

# Set NFS daemon ports
cat <<EOF>>/etc/sysconfig/nfs
QUOTAD_PORT=875
LOCKD_TCPPORT=32803
LOCKD_UDPPORT=32769
MOUNTD_PORT=892
STATD_PORT=662
STATD_OUTGOING_PORT=2020
EOF

# disable LRO on bnx2x - BZ 518531
echo "options bnx2x disable_tpa=1" >> /etc/modprobe.conf

# update software
yum -y update

# prepare action to be performed on next boot
/bin/cp /etc/rc.d/rc.local /etc/rc.d/rc.local.shipped
cat mgmt.rc.local.add >> /etc/rc.d/rc.local
) >> /root/ks-post.log 2>&1
```

1. The script that performs actions upon the boot after installation is listed below. There is some branching as the script performs different actions on each management system.

   The first management system:

   - installs python based expect commands
   - sets presentations and aliases of storage volumes for the system
   - configures the previously generated ssh onto the system
   - prepares the mount of the GFS2 file system
   - uses the other cluster member as an NTP peer
   - shuts down satellite on the temporary host
   - deploys the cluster configuration file
   - exports the LVM volume group used for management services from the temporary host
   - imports and activates the management services volume group on this system
   - starts the satellite VM on this system
   - creates the NFS volume for RHEV ISO images
   - creates the remaining Management Virtual Machines
   - starts the monitoring of the RHEV Manager

   mgmt.rc.local.add

   ```
   (
   # source env vars
   ```

```
if [[ -x varDefs.sh ]] ; then
  source varDefs.sh
elif [[ -x /root/varDefs.sh ]] ; then
  source /root/varDefs.sh
elif [[ -x /root/resources/varDefs.sh ]] ; then
  source /root/resources/varDefs.sh
elif [[ -x /root/distro/resources/varDefs.sh ]] ; then
  source /root/distro/resources/varDefs.sh
else
  echo "didn't find a varDefs.sh file"
fi

# Install CLI command libs for ILO, Storage Array,
# Virtual Console, and Onboard Administrator
/root/instpyCmds.sh /root

host=`hostname`

# Present storage LUNs to this host and configure multipath aliases
/root/prep_stor_mgmt.sh

# Wait for newly presented GFS2 disk discovery
while [ ! -e /dev/mapper/GFS2_disk ]; do sleep 2; done

# Mount temp file system and copy off prebuilt ssh configuration
/bin/mount /dev/mapper/GFS2_disk /mnt
/bin/mkdir -m 700 /root/.ssh
/bin/rm -rf /mnt/known_hosts
/bin/cp -p /mnt/* /root/.ssh
restorecon -R -v /root/.ssh
/bin/umount /mnt

# Add fstab entry for shared GFS2 volume for VM config file storage
/bin/echo "/dev/mapper/GFS2_disk /gfs2_vol  gfs2  defaults,noatime,nodiratime  0 0" >> /etc/fstab
/bin/mkdir /gfs2_vol

if [[ "$host" = "${MGMT1_NAME}" ]]
then

  # Configure the other cluster member as NTP peer
  /bin/echo "peer ${MGMT2_IP}" >> /etc/ntp.conf

  # Shutdown the satellite VM on the temporary install node and copy its
  # config file to /etc/libvirt/qemu for immediate satellite availability
  /usr/bin/ssh  ${MGMT2_IP} /usr/bin/virsh shutdown ${SAT_NAME}
  while [[ ! `/usr/bin/ssh  ${MGMT2_IP} /usr/bin/virsh list --all |grep ${SAT_NAME} |grep "shut off"`
]]; do sleep 2; done
  /usr/bin/scp ${MGMT2_IP}:/etc/libvirt/qemu/${SAT_NAME}.xml /etc/libvirt/qemu/$
{SAT_NAME}.xml
```

```
  # Deploy preconfigured cluster configuration file and set SELinux label
  /bin/mv /root/cluster.conf /etc/cluster/cluster.conf
  /sbin/restorecon /etc/cluster/cluster.conf

  # Deactivate the management services volume group on the temporary install
  # node, export from mgmt1, import to node mgmt1, and activate volume group
  /usr/bin/ssh  ${MGMT2_IP} /sbin/vgchange -an MgmtServicesVG
  /usr/bin/ssh  ${MGMT2_IP} /usr/sbin/vgexport MgmtServicesVG
  /usr/sbin/vgimport MgmtServicesVG
  /sbin/vgchange -ay MgmtServicesVG

  # Restart libvirt to recognize the management services volume group
  /sbin/service libvirtd restart
  sleep 5

  # Start the satellite VM on node mgmt1
  /usr/bin/virsh start ${SAT_NAME}

  # Create logical volume for use as RHEV NFS storage for ISO Image Library
  /usr/sbin/lvcreate -n RHEVNFSvol -L 300G MgmtServicesVG
  mkfs -t ext3 /dev/mapper/MgmtServicesVG-RHEVNFSvol
  mount /dev/mapper/MgmtServicesVG-RHEVNFSvol /mnt
  mkdir /mnt/ISO
  chown 36:36 /mnt/ISO
  umount /mnt

  # Create VMs for JON, MRG, and RHEV-M
  /root/createMgmtVMs.sh

  # Remove reference to the temporary install node in known hosts
  /bin/rm /root/.ssh/known_hosts

  # Setup RHEVM monitoring
  #  since it will modify the cluster.conf via ricci install tools and initial keys
  cd /root
  wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/riccilib-current.noarch.rpm
  wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/rhevm_mon.sh
  chmod +x /root/rhevm_mon.sh
  runcon -t rpm_t -- /usr/bin/yum -y --nogpgcheck localinstall /root/riccilib-current.noarch.rpm
  echo "*/15 * * * * /root/rhevm_mon.sh >/dev/null 2>/dev/null" >> /var/spool/cron/root

elif [[ "$host" = "${MGMT2_NAME}" ]]
then

  # Configure the other cluster member as NTP peer
  /bin/echo "peer ${MGMT1_IP}" >> /etc/ntp.conf

  # Deploy preconfigured cluster configuration file and set SELinux label
  /bin/mv /root/cluster.conf /etc/cluster/cluster.conf
  /sbin/restorecon /etc/cluster/cluster.conf
```

```
 # Add both interfaces of each node known_hosts of the other
 ssh ${MGMT1_NAME} ssh ${MGMT2_NAME}  date
 ssh ${MGMT1_NAME%.${FQD}} ssh ${MGMT2_NAME%.${FQD}}  date
 ssh ${MGMT1_IP} ssh ${MGMT2_IP}  date
 ssh ${MGMT1_ICIP} ssh ${MGMT2_ICIP} date

 # Wait for creation of JON, MRG, and RHEV-M VMs
 while [[ "`/usr/bin/ssh ${MGMT1_IP} /bin/ls /tmp/mgmtVMs 2> /dev/null`X" == "X" ]]; do sleep 5;
done

 # Shutdown standalone satellite VM for later restart as cluster service
 /usr/bin/ssh ${MGMT1_IP} /usr/bin/virsh shutdown ${SAT_NAME}
 while [[ ! `/usr/bin/ssh ${MGMT1_IP} /usr/bin/virsh list --all |grep ${SAT_NAME} |grep "shut
off"` ]]; do sleep 2; done

 # Deactivate the management service volume group
 /usr/bin/ssh ${MGMT1_IP} /sbin/vgchange -a n MgmtServicesVG

 # Set cluster locking type in LVM configuration file on both cluster members
 /usr/bin/ssh ${MGMT1_IP} "/bin/sed -i -e 's/locking_type = 1/locking_type = 3/'" /etc/lvm/lvm.conf
 /bin/sed -i -e 's/locking_type = 1/locking_type = 3/' /etc/lvm/lvm.conf

 # Configure services for clustering on both cluster members
 /usr/bin/ssh ${MGMT1_IP} /sbin/service acpid stop
 /usr/bin/ssh ${MGMT1_IP} /sbin/chkconfig acpid off
 /usr/bin/ssh ${MGMT1_IP} /sbin/chkconfig cman on
 /usr/bin/ssh ${MGMT1_IP} /sbin/chkconfig clvmd on
 /usr/bin/ssh ${MGMT1_IP} /sbin/chkconfig --level 345 gfs2 on
 /usr/bin/ssh ${MGMT1_IP} /sbin/chkconfig rgmanager on
 /sbin/service acpid stop
 /sbin/chkconfig acpid off
 /sbin/chkconfig cman on
 /sbin/chkconfig clvmd on
 /sbin/chkconfig --level 345 gfs2 on
 /sbin/chkconfig rgmanager on

 # Start the cluster services on both cluster members
 /usr/bin/ssh ${MGMT1_IP} /sbin/service cman start
 /usr/bin/ssh ${MGMT1_IP} /sbin/service clvmd start
 /sbin/service cman start
 /sbin/service clvmd start

 # Modify the management service volume group for shared cluster use and activate
 /usr/bin/ssh ${MGMT1_IP} /sbin/vgchange -c y MgmtServicesVG
 /usr/bin/ssh ${MGMT1_IP} /sbin/vgchange -a y MgmtServicesVG

 # Create a GFS2 file system for VM config file storage
 /sbin/mkfs.gfs2 -p lock_dlm -t mgmt:gfs2-1 -j 2 /dev/mapper/GFS2_disk <<-EOF
      y
```

```
        EOF

  # Mount the share GFS2 storage on both nodes and make dir for VM config files
  /usr/bin/ssh ${MGMT1_IP} /bin/mount -t gfs2 /dev/mapper/GFS2_disk /gfs2_vol/
  /bin/mount -t gfs2 /dev/mapper/GFS2_disk /gfs2_vol/
  /bin/mkdir -m 666 -p /gfs2_vol/vmconfig

  # Copy JON, MRG, and RHEV-M VM config files from mgmt1 to GFS2 shared storage
  /usr/bin/ssh ${MGMT1_IP} /bin/cp /etc/libvirt/qemu/*.xml /gfs2_vol/vmconfig
  /usr/bin/ssh ${MGMT1_IP} /bin/mv /etc/libvirt/qemu/*.xml /etc/libvirt

  # Restart libvirtd service to remove libvirt knowledge of VMs
  /usr/bin/ssh ${MGMT1_IP} /sbin/service libvirtd restart

  # Start rgmanager service once VM config files are in place on GFS2 vol
  /usr/bin/ssh ${MGMT1_IP} /sbin/service rgmanager start
  /sbin/service rgmanager start
fi

/bin/mv /etc/rc.d/rc.local /etc/rc.d/rc.local.install

# Restore original rc.local
/bin/mv /etc/rc.d/rc.local.shipped /etc/rc.d/rc.local
) >> /var/log/rc.local.out 2>&1
```

a) *instpyCmds.sh* is described in Section **6.2.1**

b) *prep_stor_mgmt.sh*:

- defines alias for each FC port on the storage array
- presents storage array LUNs to this host
- defines multipath configuration

```
#!/bin/bash

# source env vars
if [[ -x varDefs.sh ]] ; then
  echo varDefs.sh
  source varDefs.sh
elif [[ -x /root/varDefs.sh ]] ; then
  echo /root/varDefs.sh
  source /root/varDefs.sh
elif [[ -x /root/resources/varDefs.sh ]] ; then
  echo /root/resources/varDefs.sh
  source /root/resources/varDefs.sh
elif [[ -x /root/distro/resources/varDefs.sh ]] ; then
  echo /root/distro/resources/varDefs.sh
  source /root/distro/resources/varDefs.sh
else
  echo "didn't find a varDefs.sh file"
fi
```

```
#Acquire short hostname
SHORTHOST=`hostname --short`

#Modifying hostname to match name used to define host HBAs at MSA storage array
NODE=`echo $SHORTHOST | awk '{ sub("mgmt","mgmt_node"); print }'`

#Add the HBAs of this host to the MSA storage array
/root/map_fc_aliases.sh

#Present the MgmtServices and GFS2 storage volumes to each HBA in this host
for f in `sacommand --saurl //${MSA_USER}:${MSA_PW}@${MSA_IP} show hosts | grep ${SHORTHOST}_ | awk '{print $2}'`
do
  sacommand --saurl //${MSA_USER}:${MSA_PW}@${MSA_IP} map volume MgmtServices access rw ports a1,a2 lun 1 host ${f}
#  sacommand --saurl //${MSA_USER}:${MSA_PW}@${MSA_IP} map volume quorum access rw ports a1,a2 lun 2 host ${f}
  sacommand --saurl //${MSA_USER}:${MSA_PW}@${MSA_IP} map volume GFS2 access rw ports a1,a2 lun 3 host ${f}
done

#Rescan the SCSI bus to discover newly presented LUNs
/usr/bin/rescan-scsi-bus.sh

#Save original multipath configuration file
/bin/mv /etc/multipath.conf /etc/multipath.conf.orig

#Deploy multipath configuration file preconfigured with recommended
#MSA array settings for optimal performance
/bin/cp /root/multipath.conf.template /etc/multipath.conf
/root/buildMpathAliases.sh ra-msa20.cloud.lab.eng.bos.redhat.com >> /etc/multipath.conf

#Reload multipath configuration file
service multipathd reload

root@spr:/pub/projects/cloud/resources/wa/r55/resources
# vi prep_stor_mgmt.sh

root@spr:/pub/projects/cloud/resources/wa/r55/resources
# cat prep_stor_mgmt.sh
#!/bin/bash

# source env vars
if [[ -x varDefs.sh ]] ; then
  echo varDefs.sh
  source varDefs.sh
elif [[ -x /root/varDefs.sh ]] ; then
  echo /root/varDefs.sh
  source /root/varDefs.sh
```

```
elif [[ -x /root/resources/varDefs.sh ]] ; then
  echo /root/resources/varDefs.sh
  source /root/resources/varDefs.sh
elif [[ -x /root/distro/resources/varDefs.sh ]] ; then
  echo /root/distro/resources/varDefs.sh
  source /root/distro/resources/varDefs.sh
else
  echo "didn't find a varDefs.sh file"
fi

#Acquire short hostname
SHORTHOST=`hostname --short`

#Modifying hostname to match name used to define host HBAs at MSA storage array
NODE=`echo $SHORTHOST | awk '{ sub("mgmt","mgmt_node"); print }'`

#Add the HBAs of this host to the MSA storage array
/root/map_fc_aliases.sh

#Present the MgmtServices, quorum, and GFS2 storage volumes to each
#HBA in this host
for f in `sacommand --saurl //${MSA_USER}:${MSA_PW}@${MSA_IP} show hosts | grep $
{SHORTHOST}_ | awk '{print $2}'`
do
  sacommand --saurl //${MSA_USER}:${MSA_PW}@${MSA_IP} map volume MgmtServices
access rw ports a1,a2 lun 1 host ${f}
  sacommand --saurl //${MSA_USER}:${MSA_PW}@${MSA_IP} map volume GFS2 access rw
ports a1,a2 lun 3 host ${f}
done

#Enable the FC switch ports between this host and storage
vcmcommand --vcmurl //${LOGIN}:${VCM_PW}@${VCM_IP} set fc-connection $NODE 1
speed=auto
vcmcommand --vcmurl //${LOGIN}:${VCM_PW}@${VCM_IP} set fc-connection $NODE 2
speed=auto

#Rescan the SCSI bus to discover newly presented LUNs
/usr/bin/rescan-scsi-bus.sh

#Save original multipath configuration file
/bin/mv /etc/multipath.conf /etc/multipath.conf.orig

#Deploy multipath configuration file preconfigured with recommended
#MSA array settings for optimal performance
/bin/cp /root/multipath.conf.template /etc/multipath.conf
/root/buildMpathAliases.sh ${MSA_IP} >> /etc/multipath.conf

#Reload multipath configuration file
service multipathd reload
```

i) *map_fc_aliases.sh* – refer to section **6.2.1**

ii) *buildMpathAliases.sh* – refer to section **6.2.1**

c) *createMgmtVMs.sh* – This script:

- allocates LVM volumes for each Management VM
- downloads and copies Windows Image used for RHEV-M VM to correct volume
- creates a JON VM
- creates a MRG VM
- creates a RHEV-M VM
- modifies the RHEV-M VM to use the virtio network driver
- provides a CD/DVD device for each VM
- signals that the creations are complete

```
#!/bin/bash

# make volumes for Mgmt VMs
lvcreate -n JonVMvol -L 40G MgmtServicesVG
lvcreate -n MRGVMvol -L 40G MgmtServicesVG
lvcreate -n RHEVMVMvol -L 30G MgmtServicesVG

# download pre-existing compressed image for RHEV-M
wget -nv http://irish.lab.bos.redhat.com:/pub/projects/cloud/resources/rhevm.img.bz2 -O
/tmp/rhevm.img.bz2

# decompress image and copy to LVM volume
bzcat /tmp/rhevm.img.bz2 | dd of=/dev/MgmtServicesVG/RHEVMVMvol bs=1024k

# clean up compressed file
#/bin/rm /tmp/rhevm.img.bz2

# Install jon-vm
virt-install -n jon-vm -r 4096 --vcpus=2 --cpuset=auto --os-type=linux --os-variant=rhel5.4
--accelerate -w bridge:cloud0 --mac=52:54:00:C0:DE:11 --vnc --pxe --disk
path=/dev/MgmtServicesVG/JonVMvol,bus=virtio --noautoconsole --wait=-1 --noreboot

# Install mrg-vm
virt-install -n mrg-vm -r 1024 --vcpus=1 --cpuset=auto --os-type=linux --os-variant=rhel5.4
--accelerate -w bridge:cloud0 --mac=52:54:00:C0:DE:22 --vnc --pxe --disk
path=/dev/MgmtServicesVG/MRGVMvol,bus=virtio --noautoconsole --wait=-1 --noreboot

# Create RHEV-M from VM using pre-created image
virt-install -n rhevm-vm -r 4096 --vcpus=2 --cpuset=auto --os-type=windows --os-variant=win2k8
--accelerate --network=bridge:cloud0 --mac=52:54:00:C0:DE:33 --vnc --import --disk
path=/dev/MgmtServicesVG/RHEVMVMvol   --noautoconsole --wait=-1 --noreboot

# change RHEV-M network to virtio, blockio virt didn't work
sed -i "/source bridge=/a\ \ \ \ \ \ <model type=\'virtio\'/>" /etc/libvirt/qemu/rhevm-vm.xml

# insert a blank cd into each VM's definition
# jon-vm
nlines=`wc -l /etc/libvirt/qemu/jon-vm.xml | awk '{print \$1}'`
hlines=`grep -n "</disk>" /etc/libvirt/qemu/jon-vm.xml | cut -d: -f1`
tlines=`expr $nlines - $hlines`
/bin/mv /etc/libvirt/qemu/jon-vm.xml /tmp/jon-vm.xml
/usr/bin/head -n ${hlines} /tmp/jon-vm.xml > /etc/libvirt/qemu/jon-vm.xml
cat <<EOF>> /etc/libvirt/qemu/jon-vm.xml
    <disk type='file' device='cdrom'>
      <source file="/>
      <target dev='hdc' bus='ide'/>
      <readonly/>
    </disk>
EOF
/usr/bin/tail -${tlines} /tmp/jon-vm.xml >> /etc/libvirt/qemu/jon-vm.xml

# mrg-vm
nlines=`wc -l /etc/libvirt/qemu/mrg-vm.xml | awk '{print \$1}'`
hlines=`grep -n "</disk>" /etc/libvirt/qemu/mrg-vm.xml | cut -d: -f1`
tlines=`expr $nlines - $hlines`
/bin/mv /etc/libvirt/qemu/mrg-vm.xml /tmp/mrg-vm.xml
/usr/bin/head -n ${hlines} /tmp/mrg-vm.xml > /etc/libvirt/qemu/mrg-vm.xml
cat <<EOF>> /etc/libvirt/qemu/mrg-vm.xml
    <disk type='file' device='cdrom'>
      <source file="/>
      <target dev='hdc' bus='ide'/>
      <readonly/>
    </disk>
EOF
/usr/bin/tail -${tlines} /tmp/mrg-vm.xml >> /etc/libvirt/qemu/mrg-vm.xml

# rhevm-vm
nlines=`wc -l /etc/libvirt/qemu/rhevm-vm.xml | awk '{print \$1}'`
hlines=`grep -n "</disk>" /etc/libvirt/qemu/rhevm-vm.xml | cut -d: -f1`
tlines=`expr $nlines - $hlines`
/bin/mv /etc/libvirt/qemu/rhevm-vm.xml /tmp/rhevm-vm.xml
/usr/bin/head -n ${hlines} /tmp/rhevm-vm.xml > /etc/libvirt/qemu/rhevm-vm.xml
cat <<EOF>> /etc/libvirt/qemu/rhevm-vm.xml
    <disk type='file' device='cdrom'>
      <source file="/>
      <target dev='hdc' bus='ide'/>
      <readonly/>
    </disk>
EOF
/usr/bin/tail -${tlines} /tmp/rhevm-vm.xml >> /etc/libvirt/qemu/rhevm-vm.xml

# This will allow coordination with mgmt2, indicating that the
# VM creations are complete
touch /tmp/mgmtVMs
```

d) *rhevm_mon.sh* – This script is a crontab submit that:

- waits for the RHEVM to be configured
- adds the cluster monitor to the RHEVM VM entry in the cluster configuration file
- removes the crontab entry

```
#!/bin/sh

# This script is intended to be submitted as a cron job and will
# check to see if the RHEVM has been configured.  Once configured,
# add the phrase to the cluster monitor and clean the cron

# Can be submitted using a line similar to:
# echo "*/15 * * * * /root/rhevm_mon.sh >/dev/null 2>/dev/null" >> /var/spool/cron/root

# source env vars
if [[ -x varDefs.sh ]] ; then
  source varDefs.sh
elif [[ -x /root/varDefs.sh ]] ; then
  source /root/varDefs.sh
elif [[ -x /root/resources/varDefs.sh ]] ; then
  source /root/resources/varDefs.sh
elif [[ -x /root/distro/resources/varDefs.sh ]] ; then
  source /root/distro/resources/varDefs.sh
else
  echo "Didn't find a varDefs.sh file!"
fi

PATH=${PATH}:/usr/sbin

if [[ -x /usr/sbin/rhev-check.sh ]]
then

  #check status of RHEVM
  /usr/sbin/rhev-check.sh ${RHEVM_IP}
  rstat=$?

  if [[ ${rstat} -eq 0 ]]
  then
    # add RHEVM monitor to cluster
    riccicmd -H ${MGMT1_NAME} --password ${MGMT1_PW} --genkeys cluster vm $
{RHEVM_NAME} set status_program="rhev-check.sh ${RHEVM_IP}"

    # Don't need to check any longer, clear crontab
    ctLines=`wc -l /var/spool/cron/root | awk '{print $1}'`
    if [[ ${ctLines} -eq 1 ]]
    then
      crontab -r
    else
      sed -i '/rhev_mon/d' /var/spool/cron/root
    fi
  fi
```

```
else
  echo "rhev-check.sh not found!"
fi
```

6.4 Creating Management Virtual Machines

While the satellite management VM was created early in the process, the remaining management VMs are created when the first management node is installed. This section provides the details used in the creation of the JON, MRG, and RHEV-M VMs.

> NOTE: Red Hat Bugzilla 593048 may occur during any of the VM installations on KVM hosts. The logs generated during this process must be checked to confirm success.

6.4.1 JON VM

The JBoss Operations Network is the integrated management platform for JBoss instances. The following kickstart installs and configures the VM and JON software. The kickstart installs the OS, open firewall ports potentially used by JBoss/JON, and download a JON installation script which is then invoked.

jon.ks

```
install
text
key <Installation Number>
lang en_US
keyboard us
skipx
url --url http://sat-vm.cloud.lab.eng.bos.redhat.com/ks/dist/ks-rhel-x86_64-server-5-u5
network --device eth0 --bootproto static --ip 10.16.136.45 --netmask 255.255.248.0 --gateway 10.16.143.254
--nameserver 10.16.136.1,10.16.255.2 --hostname jon-vm.cloud.lab.eng.bos.redhat.com
rootpw --iscrypted $1$1o751Xnc$kmQKHj6gtZ50IILNkHkkF0
zerombr
clearpart --all --initlabel
part /boot --fstype=ext3 --size=200
part pv.01 --size=1000 --grow
part swap --size=1000   --maxsize=4000
volgroup jonvg pv.01
logvol / --vgname=jonvg --name=rootvol --size=1000 --grow
bootloader --location mbr
timezone America/New_York
auth --enablemd5 --enableshadow
selinux --permissive
poweroff
firewall --enabled
```

```
%packages

@ Base
postgresql84
postgresql84-server
java-1.6.0-openjdk.x86_64

%post
(
# MOTD
echo >> /etc/motd
echo "RHN Satellite kickstart on `date +'%Y-%m-%d'`" >> /etc/motd
echo >> /etc/motd

# Set time
ntpdate -bu 10.16.255.2
chkconfig ntpd on
/bin/cat <<EOF>/etc/ntp.conf
restrict default kod nomodify notrap nopeer noquery
restrict 127.0.0.1
driftfile /var/lib/ntp/drift
keys /etc/ntp/keys
server 10.16.136.10
server 10.16.136.15
EOF

#JBoss Required Ports
/bin/cp /etc/sysconfig/iptables /tmp/iptables
/usr/bin/head -n -2  /tmp/iptables > /etc/sysconfig/iptables
cat <<EOF>>/etc/sysconfig/iptables
-A RH-Firewall-1-INPUT -p tcp --dport 22 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 1098 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 1099 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 3873 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 4444 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 4445 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 4446 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 4457 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 8009 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 8080 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 8083 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 1100 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 1101 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 1102 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 1161 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 1162 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 3528 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 4447 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 7900 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 1102 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 1161 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 1162 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 3528 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 4447 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 7900 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 43333 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 45551 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 45556 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 45557 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 45668 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 45557 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 5432 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 67 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 68 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 7443 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 9093 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 7080 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 2098 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 2099 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 7444 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 7445 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 16163 -m state --state NEW -j ACCEPT
EOF
/usr/bin/tail -2  /tmp/iptables >> /etc/sysconfig/iptables

#register with satellite
rpm -ivh http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/rhn-org-trusted-ssl-cert-1.0-1.noarch.rpm
rhnreg_ks --activationkey=2-infraDefault --serverUrl=https://10.16.136.1/XMLRPC
--sslCACert=/usr/share/rhn/RHN-ORG-TRUSTED-SSL-CERT

#install JON software
wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/rhq-install.sh -O /tmp/rhq-install.sh
chmod 777 /tmp/rhq-install.sh
/tmp/rhq-install.sh

#update all software
/usr/bin/yum -y update
) >> /root/ks-post.log 2>&1
```

rhq-install.sh – downloaded and executed as part of the kickstart, this script installs and configures the JON software

```
#!/bin/bash
# quick 'n dirty JON/JOPR/RHQ installation/reinstallation script
# for zipped distributions
#
# Author: Steve Salevan
# Modified: BJ Lachance,  et al

# source env vars
if [[ -x varDefs.sh ]] ; then
```

```
  source varDefs.sh
elif [[ -x /root/varDefs.sh ]] ; then
  source /root/varDefs.sh
elif [[ -x /root/resources/varDefs.sh ]] ; then
  source /root/resources/varDefs.sh
elif [[ -x /root/distro/resources/varDefs.sh ]] ; then
  source /root/distro/resources/varDefs.sh
else
  echo "didn't find a varDefs.sh file"
fi

#script default values:
HOSTNAME=`hostname`
IP=`ifconfig eth0 | grep 'inet addr' | sed 's/.*inet addr://' | sed 's/ .*//'`
CURR_USER=`whoami`
AUTOINSTALL_WAITTIME=300
UNINSTALL_ONLY=0
RECREATE_USER=0

# JON installation defaults (what user gets created, where JON lands)
JON_ROOT=rhq/
JON_USER=rhq

# Java defaults

JAVA_HOME=/usr/lib/jvm/jre-openjdk

# JON-specific defaults
DB_CONNECTION_URL="jdbc:postgresql:\/\/127.0.0.1:5432\/rhq"
DB_SERVER_NAME="127.0.0.1"
HA_NAME=$HOSTNAME
SAT_SERVER=http://${SAT_FQDN}
JON_URL="$SAT_SERVER/pub/resources/jon-server-2.4.0.GA.zip"
SOA_PI_URL="$SAT_SERVER/pub/resources/jon-plugin-pack-soa-2.4.0.GA.zip"
EWS_PI_URL="$SAT_SERVER/pub/resources/jon-plugin-pack-ews-2.4.0.GA.zip"
EAP_PI_URL="$SAT_SERVER/pub/resources/jon-plugin-pack-eap-2.4.0.GA.zip"
JON_LICENSE_URL="$SAT_SERVER/pub/resources/jon-license.xml"

if [ $CURR_USER != "root" ]; then
		echo "You must be logged in as the root user to install JON."
   exit 1
fi

function jon_uninstall {
		# find service script
		echo "  * Finding JON/JOPR/RHQ service script location..."
		SVC_SCRIPT=`find /etc/init.d/ -iname "*rhq*"`
		if [ -z "$SVC_SCRIPT" ]; then
			SVC_SCRIPT=`find /etc/init.d/ -iname "*jon*"`
		fi
```

```
        if [ -z "$SVC_SCRIPT" ]; then
            echo "  - No previous installations found."
            return
        fi
        echo "  - Found JON/JOPR/RHQ service script at: $SVC_SCRIPT"

        # find home directory
        echo "  * Finding first-defined JON/JOPR/RHQ home directory..."
        for i in $SVC_SCRIPT; do
            for dir in `grep RHQ_SERVER_HOME= $i | sed 's/[-a-zA-Z0-9_]*=//'`;
            do
                    if [ -a $dir ]; then
                       JON_HOME=$dir;
                    fi
            done
            if [ -z "$JON_HOME" ]; then
                    echo "  - JON/JOPR/RHQ home directory was not defined in the service script,
 uninstall failed."
                    exit 1
            else
                    break
            fi
        done

        if [ -z "$JON_HOME" ]; then
            echo "  - JON/JOPR/RHQ home directory was not defined in the service script, uninstall
failed."
            exit 1
        fi
        echo "  - Found JON/JOPR/RHQ home directory at: $JON_HOME"

        echo "  * Stopping all services, removing service script..."
        $SVC_SCRIPT stop
        rm -f $SVC_SCRIPT

        echo "  * Dropping Postgres tables..."
        su - postgres -c "psql -c \"DROP DATABASE rhq;\""
        su - postgres -c "psql -c \"DROP USER rhqadmin;\""

        echo "  * Deleting JON/JOPR/RHQ..."
        rm -rf $JON_HOME

        echo "  - Uninstall complete!"
}

# handle CLI overrides
for i in $*
  do
  case $i in
         --jon-localuser=*)
```

```
              JON_USER="`echo $i | sed 's/[-a-zA-Z0-9]*=//'`"
              ;;
            --jon-rootdir=*)
              JON_ROOT="`echo $i | sed 's/[-a-zA-Z0-9]*=//'`"
              ;;
    --jon-url=*)
              JON_URL="`echo $i | sed 's/[-a-zA-Z0-9]*=//'`"
              ;;
            --licenseurl=*)
              JON_LICENSE_URL="`echo $i | sed 's/[-a-zA-Z0-9]*=//'`"
              ;;
            --db-connectionurl=*)
              DB_CONNECTION_URL="`echo $i | sed 's/[-a-zA-Z0-9]*=//'`"
              ;;
    --db-servername=*)
              DB_SERVER_NAME="`echo $i | sed 's/[-a-zA-Z0-9]*=//'`"
              ;;
            --ha-name=*)
              HA_NAME="`echo $i | sed 's/[-a-zA-Z0-9]*=//'`"
              ;;
            --uninstall*)
              UNINSTALL_ONLY=1
              ;;
            --recreateuser*)
              RECREATE_USER=1
              ;;
    *)
    # unknown option
            echo "You entered an option I didn't recognize."
            echo ""
            echo "If an option is not specified, a default will be used."
            echo "Available options:"
            echo "--jon-url            URL pointing to JON distribution zipfile"
            echo "--jon-localuser      Username for local user which JON will be installed under"
            echo "--jon-rootdir        Directory beneath local user's home into which JON will be installed"
            echo "--db-connectionurl   DB connection URL (e.g., jdbc:postgresql://127.0.0.1:5432/rhq)"
            echo "--db-servername      DB server name (e.g., 127.0.0.1)"
            echo "--ha-name            Name for this server, if using High Availability"
            echo "--licenseurl         URL pointing to JON license XML file"
            echo "--uninstall          Only uninstall the current JON/JOPR/RHQ instance"
            echo "--recreateuser       Create or recreate the local user account as part of installation"
            exit 1
            ;;
 esac
done

# cover uninstall only case
if [ $UNINSTALL_ONLY -eq 1 ]; then
   jon_uninstall
   exit 0
```

```
fi

# if specified JON user is not present, we must create it
/bin/egrep  -i "^$JON_USER" /etc/passwd > /dev/null
if [ $? != 0 ]; then
                echo " - Specified JON local user does not exist; hence, it will be created."
                RECREATE_USER=1
fi

# get jon and pop it into a new jon user directory
echo " * Purging any old installs and downloading JON..."
jon_uninstall

if [ $RECREATE_USER -eq 1 ]; then
                userdel -f $JON_USER
                rm -rf /home/$JON_USER
                useradd $JON_USER -p $1$1o751Xnc$kmQKHj6gtZ50IILNkHkkF0
#               mkdir /home/$JON_USER/$JON_ROOT
#               chown ${JON_USER}.${JON_USER} /home/$JON_USER/$JON_ROOT
fi

wget $SOA_PI_URL -O ./jon_soa_plugin.zip
wget $EAP_PI_URL -O ./jon_eap_plugin.zip
wget $EWS_PI_URL -O ./jon_ews_plugin.zip
wget $JON_URL -O ./jon.zip
chown $JON_USER ./jon_soa_plugin.zip
chown $JON_USER ./jon_eap_plugin.zip
chown $JON_USER ./jon_ews_plugin.zip
chown $JON_USER ./jon.zip
mv ./jon.zip /home/$JON_USER
mv ./jon_soa_plugin.zip /home/$JON_USER
mv ./jon_eap_plugin.zip /home/$JON_USER
mv ./jon_ews_plugin.zip /home/$JON_USER

# start postgres
echo " * Configuring Postgres..."
service postgresql initdb
service postgresql start

# rig postgres
su - postgres -c "psql -c \"CREATE USER rhqadmin WITH PASSWORD 'rhqadmin';\""
su - postgres -c "psql -c \"CREATE DATABASE rhq;\""
su - postgres -c "psql -c \"GRANT ALL PRIVILEGES ON DATABASE rhq to rhqadmin;\""

echo "
 # TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD

 # \"local\" is for Unix domain socket connections only
 local   all         all                               trust
 # IPv4 local connections:
```

```
 host    all         all         127.0.0.1/32          trust
 host    all         all         10.0.0.1/8         md5
 # IPv6 local connections:
 host    all         all         ::1/128               trust
" > /var/lib/pgsql/data/pg_hba.conf

chkconfig postgresql on
service postgresql restart

echo " * Unzipping and configuring JON..."
# unzip jon
su - $JON_USER -c 'unzip jon.zip'
su - $JON_USER -c 'rm jon.zip'
su - $JON_USER -c "mv jon-server-* $JON_ROOT"
su - $JON_USER -c "mv rhq* $JON_ROOT"

# configure jon's autoinstall
sed -i "s/rhq.autoinstall.enabled=false/rhq.autoinstall.enabled=true/" /home/$JON_USER/
$JON_ROOT/bin/rhq-server.properties
sed -i "s/rhq.server.high-availability.name=/rhq.server.high-availability.name=$HA_NAME/" /home/
$JON_USER/$JON_ROOT/bin/rhq-server.properties
sed -i "s/rhq.server.database.connection-
url=jdbc:postgresql:\/\/127.0.0.1:5432\/rhq/rhq.server.database.connection-
url=$DB_CONNECTION_URL/" /home/$JON_USER/$JON_ROOT/bin/rhq-server.properties
sed -i "s/rhq.server.database.server-name=127.0.0.1/rhq.server.database.server-
name=$DB_SERVER_NAME/" /home/$JON_USER/$JON_ROOT/bin/rhq-server.properties

# copy rhq-server.sh to /etc/init.d
cp /home/$JON_USER/$JON_ROOT/bin/rhq-server.sh /etc/init.d

# prepend chkconfig preamble
echo "#!/bin/sh
#chkconfig: 2345 95 20
#description: JON Server
#processname: run.sh
RHQ_SERVER_HOME=/home/$JON_USER/$JON_ROOT
RHQ_SERVER_JAVA_HOME=$JAVA_HOME" > /tmp/out
cat /etc/init.d/rhq-server.sh >> /tmp/out
mv /tmp/out /etc/init.d/rhq-server.sh
chmod 755 /etc/init.d/rhq-server.sh

# rig JON as a service
echo " * Installing JON as a service..."
chkconfig --add rhq-server.sh
chkconfig rhq-server.sh --list
chkconfig --level 3 rhq-server.sh on
chkconfig --level 5 rhq-server.sh on

# install JON license
echo " * Downloading JON license..."
```

```
wget $JON_LICENSE_URL -O /home/$JON_USER/
$JON_ROOT/jbossas/server/default/deploy/rhq.ear.rej/license/license.xml

echo " * Starting JON for the first time..."
service rhq-server.sh start

# install JON plugins
echo " * Waiting until server installs..."
sleep $AUTOINSTALL_WAITTIME #wait for autoinstall to finish

echo " * Installing plugins..."
for i in /home/$JON_USER/*.zip ; do unzip -d /tmp/rhq-plugins $i *.jar ; done
find /tmp/rhq-plugins -name "*.jar" | xargs -i[] mv [] /home/$JON_USER/
$JON_ROOT/jbossas/server/default/deploy/rhq.ear/rhq-downloads/rhq-plugins
/bin/rm -rf /tmp/rhq-plugins

echo " * Restarting JON..."
service rhq-server.sh stop
service rhq-server.sh start
```

6.4.2 MRG VM

The MRG VM kickstart installs the VM and configures MRG settings. The kickstart installs the OS, opens the required firewall ports, adds the MRG Manager user, enables the required services, and configures actions to be performed upon the next boot of the VM.

mrggrid.ks

```
install
text
network --device eth0 --bootproto static --ip 10.16.136.50 --netmask 255.255.248.0 --gateway 10.16.143.254
--nameserver 10.16.136.1,10.16.255.2 --hostname mrg-vm.cloud.lab.eng.bos.redhat.com
url --url http://sat-vm.cloud.lab.eng.bos.redhat.com/ks/dist/ks-rhel-x86_64-server-5-u5
lang en_US
keyboard us
zerombr
clearpart --all
part /boot --fstype=ext3 --size=200
part pv.01 --size=1000 --grow
part swap --size=1000   --maxsize=2000
volgroup MRGVG pv.01
logvol / --vgname=MRGVG --name=rootvol --size=1000 --grow
bootloader --location mbr
timezone America/New_York
auth --enablemd5 --enableshadow
rootpw --iscrypted $1$1o751Xnc$kmQKHj6gtZ50IILNkHkkF0
selinux --permissive
poweroff
firewall --enabled
skipx
```

```
key --skip
user --name=admin --password=$1$1o751Xnc$kmQKHj6gtZ50IILNkHkkF0 --iscrypted

%packages

@ Base
pexpect
ntp
SDL
ruby

%post
(
# MOTD
echo >> /etc/motd
echo "RHN Satellite kickstart on `date +'%Y-%m-%d'`" >> /etc/motd
echo >> /etc/motd

ntpdate -bu 10.16.255.2
chkconfig ntpd on
/bin/cat <<EOF>/etc/ntp.conf
restrict default kod nomodify notrap nopeer noquery
restrict 127.0.0.1
driftfile /var/lib/ntp/drift
keys /etc/ntp/keys
server 10.16.136.10
server 10.16.136.15
EOF

# register system with satellite
rpm -ivh http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/rhn-org-trusted-ssl-cert-1.0-1.noarch.rpm
rhnreg_ks --activationkey=2-infraMRGGrid --serverUrl=https://10.16.136.1/XMLRPC
--sslCACert=/usr/share/rhn/RHN-ORG-TRUSTED-SSL-CERT
rhn_check

#Configure iptables
/bin/cp /etc/sysconfig/iptables /tmp/iptables
/usr/bin/head -n -2  /tmp/iptables > /etc/sysconfig/iptables
cat <<EOF>>/etc/sysconfig/iptables
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 875 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 875 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 892 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 892 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 32769 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 32769 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 32803 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 32803 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 662 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 662 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 111 -j ACCEPT
```

```
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 111 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 2049 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 2049 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 2020 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 2020 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 4672 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 4672 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 5672 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 5672 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 45672 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 45672 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 9618 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 9618 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 9614 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 9614 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 9600:9800 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 9600:9800 -m state --state NEW -j ACCEPT
EOF
/usr/bin/tail -2  /tmp/iptables >> /etc/sysconfig/iptables

#tie down nfs ports
cat <<EOF>>/etc/sysconfig/nfs
RQUOTAD_PORT=875
LOCKD_TCPPORT=32803
LOCKD_UDPPORT=32769
MOUNTD_PORT=892
STATD_PORT=662
STATD_OUTGOING_PORT=2020
EOF

# setup export for rendering
mkdir /home/admin/render
chown admin:admin /home/admin/render
cat <<EOF>>/etc/exports
/home/admin/render *.cloud.lab.eng.bos.redhat.com(rw,sync,no_root_squash)
EOF

#Add mrgmgr user
#useradd mrgmgr

#Turn on Services
chkconfig sesame on
chkconfig postgresql on
chkconfig condor on
chkconfig qpidd on
#chkconfig cumin on
chkconfig ntpd on
chkconfig nfs on

#Postgresql funkiness
```

```
rm -rf /var/lib/pgsql/data
su - postgres -c "initdb -D /var/lib/pgsql/data"

# update software
yum -y update

#prepare actions to be performed on next boot
wget http://10.16.136.1/pub/resources/mrggrid.rc.local.add -O /etc/rc.d/mrggrid.rc.local.add
/bin/cp /etc/rc.d/rc.local /etc/rc.d/rc.local.shipped
cat  /etc/rc.d/mrggrid.rc.local.add >> /etc/rc.d/rc.local
) >> /root/ks-post.log 2>&1
```

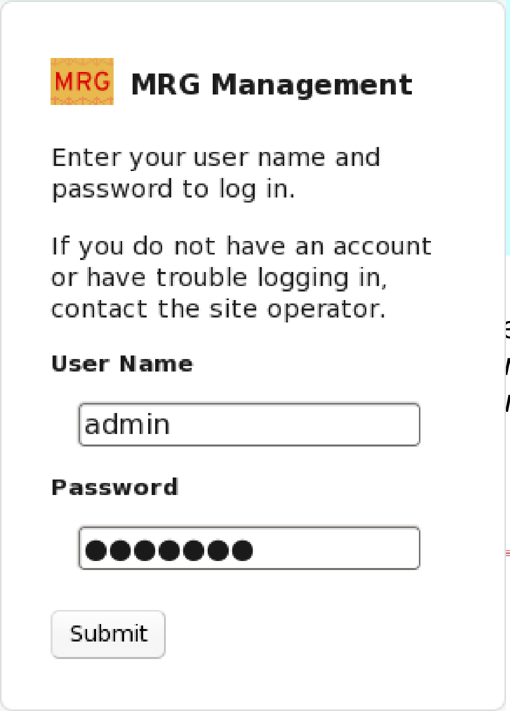
mrggrid.rc.local.add – This script performs the actions to be performed upon the next boot of the MRG VM include downloading configuration files and scripts for MRG Manager, initializing the database, creating the admin user, and restarting the cumin service.

```
(

echo "--- Setting up render job ---"

#populate rendering data
wget -nv http://irish.lab.bos.redhat.com/pub/projects/cloud/resources/render.tgz -O /tmp/render.tgz
tar -xpzf /tmp/render.tgz -C /home/admin
/bin/rm /tmp/render.tgz

# make blender available
wget -nv http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/blender-2.48a-linux-glibc236-py24-
x86_64-static.tar.bz2 -O /tmp/blender-2.48a-linux-glibc236-py24-x86_64-static.tar.bz2
su - admin -c "bzcat /tmp/blender-2.48a-linux-glibc236-py24-x86_64-static.tar.bz2 | tar xf -"

echo "--- Setting up perfect number search job ---"
# Get configuration and script files  used by MRG Grid Manager
echo "--- Retrieving files ---"
wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/cumin.conf -O /etc/cumin/cumin.conf
wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/pg_hba.conf -O
/var/lib/pgsql/data/pg_hba.conf
chown postgres.postgres /var/lib/pgsql/data/pg_hba.conf
chmod 600 /var/lib/pgsql/data/pg_hba.conf
wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/condor_config.local.mgr -O
/var/lib/condor/condor_config.local
restorecon -R -v /var/lib

#Initialize the database
echo "--- initialize cumin ---"
cumin-database-init

#Add the admin user
echo "--- Add cumin user ---"
wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/add_cumin_user.py -O
/tmp/add_cumin_user.py
chmod +x /tmp/add_cumin_user.py
/tmp/add_cumin_user.py admin 24^gold

#Start Cumin
echo "--- Starting cumin ---"
chkconfig cumin on
service cumin start

#Restore original rc.local
echo "--- Putting original rc.local back ---"
/bin/mv /etc/rc.d/rc.local /etc/rc.d/rc.local.install
/bin/mv /etc/rc.d/rc.local.shipped /etc/rc.d/rc.local

#setup the perfect number application
wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/perfect.tgz -O /tmp/perfect.tgz
tar -xpzf /tmp/perfect.tgz -C /home/admin
/bin/mv /home/admin/perfect/perfect.py /usr/tmp/perfect.py
/bin/mv /home/admin/perfect/perfect /usr/tmp/perfect

echo COMPLETE

)  >> /var/log/rc.local.out 2>&1
```

add_cumin_user.py – this script, called from *rc.local*, is a small expect script to add the cumin user.

```
#!/usr/bin/python
import sys, os, pexpect

cuser = sys.argv[1]
passwd = sys.argv[2]

signproc = pexpect.spawn("cumin-admin add-user %s" % cuser)
match = signproc.expect(["Enter new password:"])

signproc.sendline(p

match = signproc.e

signproc.sendline(p

match = signproc.e
```

The configuration of the MRG e user to open the appropriate URL (e.g., *http://r m:45672/login.html*) for logging in and providing the e *rd.*



MRG Management

Enter your user name and password to log in.

If you do not have an account or have trouble logging in, contact the site operator.

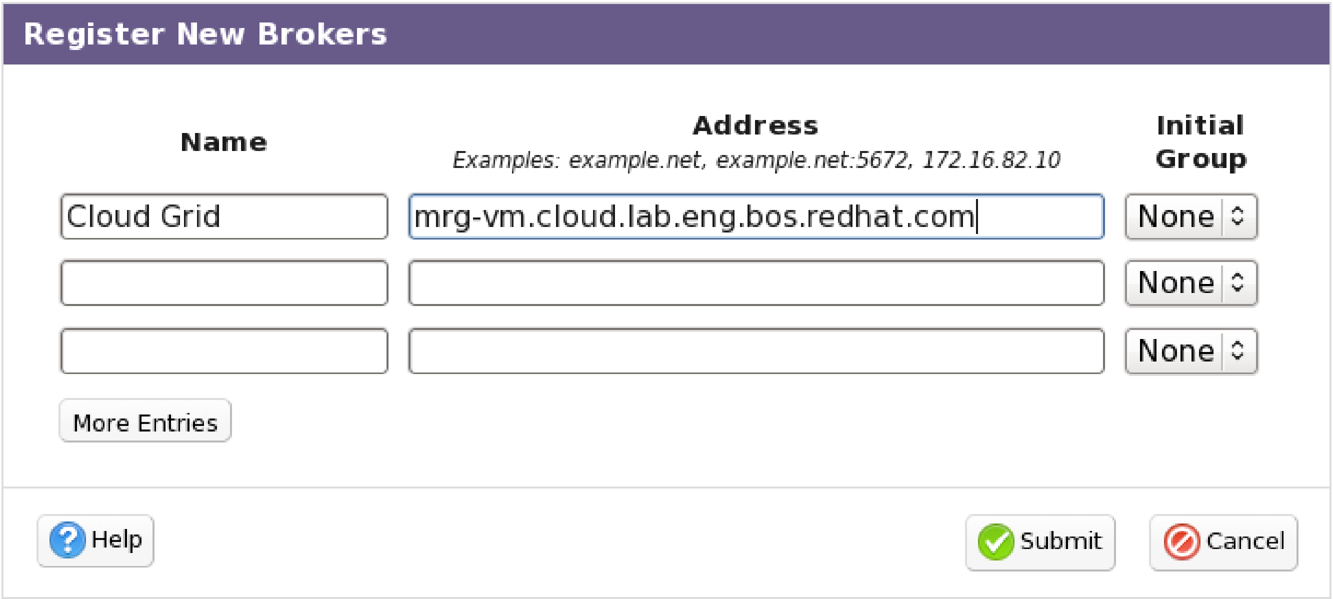
User Name

admin

Password

●●●●●●●

In the *Home* tab, click the *Add Management Data Source* button. Supply a *Name* and the *Address* of the MRG VM, then select the *Submit* button.



Register New Brokers

Name	Address Examples: example.net, example.net:5672, 172.16.82.10	Initial Group
Cloud Grid	mrg-vm.cloud.lab.eng.bos.redhat.com	None
		None
		None

6.4.3 RHEV-M VM

Because RHEV-M uses Windows as a platform, the media is not supplied by Red Hat. The VM is created using a previously generated image (refer to Appendix **A.3**) that is dropped into place. However, this is not required and a blank VM may be created to allow an operator to install using their own media. The procedure to create the image used is detailed in Appendix **A.3**. The image itself is not completely configured for use. Refer to Section **6.7** for the actions required to complete the configuration of this VM.

6.5 Provision Second Management Node

When the satellite VM boots for the second time, it starts the installation of the second management node. This occurs during the first management node's post installation when it shuts down satellite on the temporary host, performs some preparatory work, and restarts the VM on itself.

The contents of *sat.rc.local2.add* are appended to */etc/rc.local* to specify the commands invoked on the second boot of the satellite VM.

sat.rc.local2.add

```
(
/bin/echo "Performing 2nd round rc.local commands"

# install 2nd mgmt node
/root/resources/instMgmt2.sh

/bin/cat /etc/rc.d/rc.local.shipped /root/resources/sat.rc.local3.add > /etc/rc.d/rc.local
) >> /var/log/rc.local2.out 2>&1
```

The *instMgmt2.sh* script:

- adds cobbler system entry for system
- sets boot order so PXE is first
- powers on/reset blade
- waits before setting PXE boot to last

```
#!/bin/bash

# source env vars
if [[ -x varDefs.sh ]] ; then
  source varDefs.sh
elif [[ -x /root/varDefs.sh ]] ; then
  source /root/varDefs.sh
elif [[ -x /root/resources/varDefs.sh ]] ; then
  source /root/resources/varDefs.sh
elif [[ -x /root/distro/resources/varDefs.sh ]] ; then
  source /root/distro/resources/varDefs.sh
else
  echo "didn't find a varDefs.sh file"
fi

#prepare cobbler to boot this node
cobbler system add --name=$MGMT2_NAME --profile=$MGMT_PROFILE --mac=$MGMT2_MAC
--ip=$MGMT2_IP --hostname=$MGMT2_NAME --ksmeta="NAME=$MGMT2_NAME
NETIP=$MGMT2_IP ICIP=$MGMT2_ICIP ICNAME=$MGMT2_IC" –
kopts="console=ttyS0,115200 nostorage"
cobbler sync

# set to boot PXE
while [[ ! `ilocommand -i //${LOGIN}:${MGMT2_ILO_PW}@${MGMT2_ILO} set
/system1/bootconfig1/bootsource5 bootorder=1 | grep status=0` ]]; do sleep 2; done

#reset blade, power up also in case it was powered off
ilocommand -i //${LOGIN}:${MGMT2_ILO_PW}@${MGMT2_ILO} power reset
ilocommand -i //${LOGIN}:${MGMT2_ILO_PW}@${MGMT2_ILO} power on

#wait a reasonable amount of time and change the boot order to make network last
sleep 240
while [[ ! `ilocommand -i //${LOGIN}:${MGMT2_ILO_PW}@${MGMT2_ILO} set
/system1/bootconfig1/bootsource5 bootorder=5 | grep status=0` ]]; do sleep 2; done
```

Because the same kickstart used for the first management system is used on the second, the same activities occur during install. Upon reboot however, the preconfigured activities branch significantly depending on the host upon which they execute. The script is the same and is referenced in the previous section. The actions taken by the second node:

- install python based expect commands
- set presentations and aliases of storage volumes for the system

- configure the previously generated ssh onto the system
- prepare the mount of the GFS2 file system
- use the other cluster member an NTP peer
- deploy the cluster configuration file
- execute round robin ssh connections to create secure shell known hosts
- await a signal from the first node that it has created the management VMs
- shut down satellite on the first cluster node
- configure LVM volume for cluster use
- start the CMAN and CLVMD cluster services
- create and mount the GFS2 file system
- place the VM definition files onto the GFS2 file system
- complete starting the cluster daemons which then start all the services, including the management VMs

6.6 Create First Hosts

When the satellite VM boots for the third time, it creates the first of each of the RHEL/KVM and RHEV hypervisor hosts. This third boot occurs after the cluster has been formed when the satellite VM is started as a cluster service.

sat.rc.local3.add

```
(
# source env vars
if [[ -x varDefs.sh ]] ; then
  source varDefs.sh
elif [[ -x /root/varDefs.sh ]] ; then
  source /root/varDefs.sh
elif [[ -x /root/resources/varDefs.sh ]] ; then
  source /root/resources/varDefs.sh
elif [[ -x /root/distro/resources/varDefs.sh ]] ; then
  source /root/distro/resources/varDefs.sh
else
  echo "didn't find a varDefs.sh file"
fi

# prime the riccicmd
riccicmd -v -H ${MGMT1_IP} --password ${MGMT1_PW} --genkeys cluster configuration

# install first RHEVH host
/root/resources/instRHEVH.sh ${RHEVH01_VC_PROFILE} &

# install first RHELH host
# delay RHEL host install due to BZ 602402
sleep 3600
/root/resources/instRHELH.sh ${RHELH01_VC_PROFILE} &

/bin/mv /etc/rc.d/rc.local.shipped /etc/rc.d/rc.local
) >> /var/log/rc.local3.out 2>&1
```

1. The *instRHELH.sh* script requires a single passed parameter, the server blade profile name assigned in the HP Virtual Connect interface. The kickstart for this installation is described in Section **6.6.1** . The script:
   - retrieves all identity information
   - creates the cobbler system entry
   - presents storage to the host
   - adds the new NFS export stanza to the cluster configuration file
   - sets the boot order to boot PXE first
   - registers the host with satellite after install
   - sets the boot order to boot PXE last

```
#!/bin/bash
```

```
# This script will install and prepare a system to be a RHEL host

# Source env vars
if [[ -x varDefs.sh ]] ; then
  source varDefs.sh
elif [[ -x /root/varDefs.sh ]] ; then
  source /root/varDefs.sh
elif [[ -x /root/resources/varDefs.sh ]] ; then
  source /root/resources/varDefs.sh
elif [[ -x /root/distro/resources/varDefs.sh ]] ; then
  source /root/distro/resources/varDefs.sh
else
  echo "didn't find a varDefs.sh file"
fi

# The blade profile must be passed
if [[ $# -ne 1 ]]
then
  echo 'Usage - $0 <HP Virtual Connect profile name>'
  exit -1
else
  pname=$1
fi

vcmcommand --vcmurl //${LOGIN}:${VCM_PW}@${VCM_IP} show server | grep ${pname} >
/dev/null 2>/dev/null
if [[ $? -ne 0 ]]
then
  echo "HP Virtual Connect profile $pname not found!"
  exit -2
fi

nblade=`vcmcommand --vcmurl //${LOGIN}:${VCM_PW}@${VCM_IP} show server | grep $
{pname} | awk '{print $3}'`

iloIP=`oacommand --oaurl //${LOGIN}:${OA_PW}@${OA_IP} show server info ${nblade} | grep
"IP Address" | awk '{print $3}'`

rawMAC=`vcmcommand --vcmurl //${LOGIN}:${VCM_PW}@${VCM_IP} show profile ${pname} |
grep public | awk '{print $5}'`

#implement a semaphore to verify name and IP are unique
while [[ -e /tmp/AvailNameIP ]] ; do sleep 1; done
touch /tmp/AvailNameIP

IPname=`/root/resources/GetAvailRhelh.sh | awk '{print $1}'`

IPnum=`/root/resources/GetAvailRhelh.sh | awk '{print $2}'`

# Create cobbler system entry
```

```
echo -e "\nCreating cobbler system entry ...\n"
cobbler system add --name=${IPname} --profile=${RHELH_PROFILE} --mac=${rawMAC//-/:}
--ip=${IPnum} --hostname=${IPname} --dns-name=${IPname} --kopts="console=ttyS0,115200
nostorage"
cobbler sync

#Remove semaphore
/bin/rm /tmp/AvailNameIP

# Present storage to hosts
echo -e "\nPresenting storage to host ${pname} ...\n"
/root/resources/prep_stor_host.sh ${pname}

# Update cluster configuration for NFS presentation

# create a semaphore for unique client names
while [[ -e /tmp/UpdateNFSClient ]] ; do sleep 1; done
touch /tmp/UpdateNFSClient

#Get next available client name
nfsClient=`/root/resources/GetAvailNFSClient.sh`

# add the export to the cluster configuration
addnfsexport --ricciroot=/.ricci -H ${MGMT1_IP} rhev-nfs-fs ${nfsClient} ${IPname} rw 1

# release semaphore
/bin/rm /tmp/UpdateNFSClient

# Get the count of systems registered with this name (should be 0)
initReg=`/root/resources/listRegSystems_infra.py | grep -c ${IPname}`
echo -e "\nNumber of systems registered with this name: ${initReg} ...\n"

# Set to boot PXE first
echo -e "\nChanging system boot order to boot network (PXE) first ...\n"
while [[ ! `ilocommand -i //${LOGIN}:${ILO_PW}@${iloIP} set /system1/bootconfig1/bootsource5
bootorder=1 | grep status=0` ]]; do sleep 10; done

# Reset node (power on node in case node was off)
echo -e "\nResetting server blade (power on in case blade was off) ...\n"
ilocommand -i //${LOGIN}:${ILO_PW}@${iloIP} power reset
ilocommand -i //${LOGIN}:${ILO_PW}@${iloIP} power on

# Wait for system to register with satellite indicating installation completion
echo -e "\nWaiting for system to register with satellite ...\n"
while [[ $initReg -ge `/root/resources/listRegSystems_infra.py | grep -c ${IPname}` ]]; do sleep 15;
done
echo -e "\nSatellite registration complete ...\n"

# Set to boot PXE last
echo -e "\nChanging system boot order to boot network last ...\n"
```

```
while [[ ! `ilocommand -i //${LOGIN}:${ILO_PW}@${iloIP} set /system1/bootconfig1/bootsource5
bootorder=5 | grep status=0` ]]; do sleep 2; done
```

a) In addition to using several of the *commands and addnfsexport (part of riccicmd), described in Section **6.1.1** , other scripts and commands are called from within the above script. The first of which is *GetAvailRhelh.sh* which returns the next available IP name and address that can be used for a RHEL/KVM host.

```
#!/bin/bash

# This script provides an available RHELH hostname and IP address.

# Source env vars
if [[ -x varDefs.sh ]] ; then
  source varDefs.sh
elif [[ -x /root/varDefs.sh ]] ; then
  source /root/varDefs.sh
elif [[ -x /root/resources/varDefs.sh ]] ; then
  source /root/resources/varDefs.sh
elif [[ -x /root/distro/resources/varDefs.sh ]] ; then
  source /root/distro/resources/varDefs.sh
else
  echo "didn't find a varDefs.sh file"
fi

HOSTS=`host -l ${FQD} ${SAT_FQDN} |grep rhelh | cut -f1 -d'.' | sort`
if [[ ${HOSTS} ]]
then
  # Some hosts exist
  indx=1
  thost=`printf "rhelh-%02d" ${indx}`
  while [[ `echo ${HOSTS} | grep ${thost}` ]]
  do
    indx=`expr $indx + 1`
    thost=`printf "rhelh-%02d" ${indx}`
  done
  fhost="${thost}.${FQD}"
else
  # No existing hosts
  fhost="rhelh-01.${FQD}"
fi

# Find an available IP

# Assumes Satellite uses the lowest IP address
IP_DOMAIN=${SAT_IP%.*}
indx=1
tIP=`printf "%s.%d" ${IP_DOMAIN} ${indx}`
host ${tIP} > /dev/null 2>/dev/null
while [[  $? -eq 0 ]]
do
```

```
  indx=`expr $indx + 1`
  tIP=`printf "%s.%d" ${IP_DOMAIN} ${indx}`
  host ${tIP} > /dev/null 2>/dev/null
done

echo "${fhost} ${tIP}"
```

b) *prep_stor_host.sh* – adds/verifies each of the host's HBA, then presents the LUN used for RHEV to all the host's HBAs

```
#!/bin/bash

#source variables
if [[ -x varDefs.sh ]]
then
  source varDefs.sh
elif [[ -x /root/varDefs.sh ]]
then
  source /root/varDefs.sh
elif [[ -x /root/resources/varDefs.sh ]]
then
  source /root/resources/varDefs.sh
elif [[ -x /root/distro/resources/varDefs.sh ]]
then
  source /root/distro/resources/varDefs.sh
else
  echo "didn't find a varDefs.sh file"
fi


# Confirm that a parameter was passed
if [[ $# -ne 1 ]]
then
   echo "Usage: $0 <profile name>\n"
   exit -1
fi

PNAME=$1

# Confirm that the passed is an existing profile
if [[ ! `vcmcommand --vcmurl //${LOGIN}:${VCM_PW}@${VCM_IP} show profile | grep $
{PNAME}` ]]
then
  echo ${PNAME} " is not an existing profile!"
  exit -2
fi

#Add the HBAs of this host to the MSA storage array
if [[ `sacommand --saurl //${MSA_USER}:${MSA_PW}@${MSA_IP} show hosts | grep $
{PNAME}_` ]]
then
```

```
  echo "Aliases for this host already exist!"
else
  vcmcommand --vcmurl //${LOGIN}:${VCM_PW}@${VCM_IP} show profile ${PNAME} |
grep -A 6 "FC SAN Connections" | grep "[0-9]\+\\W\+[0-9]\+" | awk '{print $1 " " $6}' | while read
PORT WWN
  do
    # if storage see HBA set the nickname, otherwise create host entry
    if [[ `sacommand --saurl //${MSA_USER}:${MSA_PW}@${MSA_IP} show hosts | grep  $
{WWN//:}` ]]
    then
      sacommand --saurl //${MSA_USER}:${MSA_PW}@${MSA_IP} set host-name id $
{WWN//:} ${PNAME}_${PORT}
    else
      sacommand --saurl //${MSA_USER}:${MSA_PW}@${MSA_IP} create host id ${WWN//:}
${PNAME}_${PORT}
    fi
  done
fi

#Present the RHEVStorage1 storage volumes to each HBA in host
for f in `sacommand --saurl //${MSA_USER}:${MSA_PW}@${MSA_IP} show hosts | grep $
{PNAME}_ | awk '{print $2}'`
do
  sacommand --saurl //${MSA_USER}:${MSA_PW}@${MSA_IP} map volume
msa20_RHEVStorage1 access rw ports a1,a2 lun 4 host ${f}
done
```

c) *GetAvailNFSClient.sh* – output the next unique identifier for the NFS client tag in the cluster configuration file

```
#!/bin/bash

# This script will provide the an available nfs-client idenitifier

# Source env vars
if [[ -x varDefs.sh ]] ; then
  source varDefs.sh
elif [[ -x /root/varDefs.sh ]] ; then
  source /root/varDefs.sh
elif [[ -x /root/resources/varDefs.sh ]] ; then
  source /root/resources/varDefs.sh
elif [[ -x /root/distro/resources/varDefs.sh ]] ; then
  source /root/distro/resources/varDefs.sh
else
  echo "didn't find a varDefs.sh file"
fi

nfsClients=`riccicmd --ricciroot=/.ricci -H ${MGMT1_IP} cluster configuration | grep nfsclient  |
grep name= | awk '{print $3}' | cut -f2 -d\" | sort -u`
if [[ ${nfsClients} ]]
then
```

```
  indx=1
  tname=`printf "rhev-nfs-client-%02d" ${indx}`
  while [[ `echo ${nfsClients} | grep ${tname}` ]]
  do
    indx=`expr $indx + 1`
    tname=`printf "rhev-nfs-client-%02d" ${indx}`
  done
else
  # No existing hosts
  tname="rhev-nfs-client-01"
fi

echo ${tname}
```

d) *listRegSystems_infra.py* – list system registered to the RHN Satellite system

```
#!/usr/bin/python
"""
This script lists all registered systems.
The org is determined by the login global variable below.
"""

import xmlrpclib
import sys


SATELLITE_URL = "http://sat-vm.cloud.lab.eng.bos.redhat.com/rpc/api"


INFRA_LOGIN = "infra"
INFRA_PASSWD = "24^gold"


#open channel
client = xmlrpclib.Server(SATELLITE_URL, verbose=0)

#log into infrastructure org
key = client.auth.login(INFRA_LOGIN, INFRA_PASSWD)

aSYS = client.system.listSystems(key)
for iSYS in aSYS:
    print iSYS['name']

#log out from infrastructure channel
client.auth.logout(key)
```

2. The install script *instRHEVH.sh* installs and prepares a system for use as a RHEV host. It requires a single passed parameter, the server blade profile name assigned in the HP Virtual Connect interface, and:
    - retrieves all identity information and creates the cobbler system entry
    - presents storage to the host

- adds the new NFS export stanza to the cluster configuration file
- sets the boot order to boot PXE first
- registers the host with satellite after install
- sets the boot order to boot PXE last

```
#!/bin/bash
#
# This script will install and prepare a system for use as a RHEV host

# source env vars
if [[ -x varDefs.sh ]] ; then
  source varDefs.sh
elif [[ -x /root/varDefs.sh ]] ; then
  source /root/varDefs.sh
elif [[ -x /root/resources/varDefs.sh ]] ; then
  source /root/resources/varDefs.sh
elif [[ -x /root/distro/resources/varDefs.sh ]] ; then
  source /root/distro/resources/varDefs.sh
else
  echo "Didn't find a varDefs.sh file!"
fi

# The blade profile must be passed
if [[ $# -ne 1 ]]
then
  echo 'Usage - $0 <HP Virtual Connect profile name>'
  exit -1
else
  pname=$1
fi

# Implement a semaphore to verify that name and IP are unique
while [[ -e /tmp/AvailNameIP ]] ; do sleep 1; done
touch /tmp/AvailNameIP

vcmcommand --vcmurl //${LOGIN}:${VCM_PW}@${VCM_IP} show server | grep ${pname}
> /dev/null 2>/dev/null
if [[ $? -ne 0 ]]
then
  echo "HP Virtual Connect profile $pname not found!"
  exit -2
fi

nblade=`vcmcommand --vcmurl //${LOGIN}:${VCM_PW}@${VCM_IP} show server | grep $
{pname} | awk '{print $3}'`
iloIP=`oacommand --oaurl //${LOGIN}:${OA_PW}@${OA_IP} show server info ${nblade} |
grep "IP Address" | awk '{print $3}'`
rawMAC=`vcmcommand --vcmurl //${LOGIN}:${VCM_PW}@${VCM_IP} show profile $
{pname} | grep public | awk '{print $5}'`
IPname=`/root/resources/GetAvailRhevh.sh | awk '{print $1}'`
```

```
IPnum=`/root/resources/GetAvailRhevh.sh | awk '{print $2}'`

# Delete any previously defined cobbler system entry
if [[ `cobbler system list | grep ${IPname}` ]]
then
  cobbler system delete --name=${IPname}
fi

# Create cobbler system entry
echo -e "\nCreating cobbler system entry ...\n"
cobbler system add --name=${IPname} --profile=${RHEVH_PROFILE} --mac=${rawMAC//-/:}
--ip=${IPnum} --hostname=${IPname} --dns-name=${IPname} --kopts="console=ttyS0,115200
nostorage"
cobbler sync

# Remove semaphore
/bin/rm /tmp/AvailNameIP

# Present storage to host
echo -e "\nPresenting storage to host ${pname} ...\n"
/root/resources/prep_stor_host.sh ${pname}

# Update cluster configuration for NFS presentation

# create a semaphore for unique client names
while [[ -e /tmp/UpdateNFSClient ]] ; do sleep 1; done
touch /tmp/UpdateNFSClient

#Get next available client name
nfsClient=`/root/resources/GetAvailNFSClient.sh`

# add the export to the cluster configuration
addnfsexport --ricciroot=/.ricci -H ${MGMT1_IP} rhev-nfs-fs ${nfsClient} ${IPname} rw 1

# release semaphore
/bin/rm /tmp/UpdateNFSClient

# Get the count of systems registered with this name (should be 0)
initReg=`/root/resources/listRegSystems_infra.py | grep -c ${IPname}`
echo -e "\nNumber of systems registered with this name: ${initReg} ...\n"

# Change system boot order to boot network (PXE) first
echo -e "\nChanging system boot order to boot network (PXE) first ...\n"
while [[ ! `ilocommand -i //${LOGIN}:${ILO_PW}@${iloIP} set
/system1/bootconfig1/bootsource5 bootorder=1 | grep status=0` ]]; do sleep 10; done

# Reset server blade (power on in case blade was off)
echo -e "\nResetting server blade (power on in case blade was off) ...\n"
ilocommand -i //${LOGIN}:${ILO_PW}@${iloIP} power reset
ilocommand -i //${LOGIN}:${ILO_PW}@${iloIP} power on
```

```
# Wait for system to register with satellite indicating installation completion
echo -e "\nWaiting for system to register with satellite ...\n"
while [[ $initReg -ge `/root/resources/listRegSystems_infra.py | grep -c ${IPname}` ]]; do sleep 5;
done
echo -e "\nSatellite registration complete ...\n"

# Change system boot order to boot network last
echo -e "\nChanging system boot order to boot network last ...\n"
while [[ ! `ilocommand -i //${LOGIN}:${ILO_PW}@${iloIP} set
/system1/bootconfig1/bootsource5 bootorder=5 | grep status=0` ]]; do sleep 2; done
```

a) Similar to *instRHELH.sh*, the above script calls many of the same scripts. However, *GetAvailRhevh.sh* is unique, returning the next available IP name and address that are assigned to the RHEV-H host.

```
#!/bin/bash

# This script provides an available RHELH hostname and IP address.

# Source env vars
if [[ -x varDefs.sh ]] ; then
  source varDefs.sh
elif [[ -x /root/varDefs.sh ]] ; then
  source /root/varDefs.sh
elif [[ -x /root/resources/varDefs.sh ]] ; then
  source /root/resources/varDefs.sh
elif [[ -x /root/distro/resources/varDefs.sh ]] ; then
  source /root/distro/resources/varDefs.sh
else
  echo "didn't find a varDefs.sh file"
fi

HOSTS=`host -l ${FQD} ${SAT_FQDN} |grep rhevh | cut -f1 -d'.' | sort`
if [[ ${HOSTS} ]]
then
  # Some hosts exist
  indx=1
  thost=`printf "rhevh-%02d" ${indx}`
  while [[ `echo ${HOSTS} | grep ${thost}` ]]
  do
    indx=`expr $indx + 1`
    thost=`printf "rhevh-%02d" ${indx}`
  done
  fhost="${thost}.${FQD}"
else
  # No existing hosts
  fhost="rhevh-01.${FQD}"
fi

# Find an available IP
# Assumes Satellite uses the lowest IP address
IP_DOMAIN=${SAT_IP%.*}
indx=1
tIP=`printf "%s.%d" ${IP_DOMAIN} ${indx}`
host ${tIP} > /dev/null 2>/dev/null
while [[  $? -eq 0 ]]
do
  indx=`expr $indx + 1`
  tIP=`printf "%s.%d" ${IP_DOMAIN} ${indx}`
  host ${tIP} > /dev/null 2>/dev/null
done

echo "${fhost} ${tIP}"
```

6.6.1 RHEL Configuration for a RHEV Host

When the RHEL/KVM host is started for the first time, the kickstart installs the system with the base software including NTP. Additionally, the post-install script:

- retrieves the time from a trusted source and sets the platform hardware clock
- configures the firewall to work with RHEV
- enables the RHN service
- disables LRO
- updates all software
- works around known issues which include a host reboot

rhelHost.ks

```
install
text
network --device eth0 --noipv6 --bootproto dhcp
network --device eth1 --noipv6 --onboot no --bootproto dhcp
network --device eth2 --noipv6 --onboot no --bootproto dhcp
network --device eth3 --noipv6 --onboot no --bootproto dhcp
network --device eth4 --noipv6 --onboot no --bootproto dhcp
network --device eth5 --noipv6 --onboot no --bootproto dhcp
network --device eth6 --noipv6 --onboot no --bootproto dhcp
network --device eth7 --noipv6 --onboot no --bootproto dhcp
url --url http://sat-vm.cloud.lab.eng.bos.redhat.com/ks/dist/ks-rhel-x86_64-server-5-u5
lang en_US
keyboard us
device scsi cciss
zerombr
clearpart --linux --drives=cciss/c0d0
part /boot --fstype=ext3 --size=200 --ondisk=cciss/c0d0
part pv.01 --size=1000 --grow --ondisk=cciss/c0d0
part swap --size=10000   --maxsize=20000 --ondisk=cciss/c0d0
volgroup rhelh_vg pv.01
logvol / --vgname=rhelh_vg --name=rootvol --size=1000 --grow
bootloader --location mbr
```

```
timezone America/New_York
auth --enablemd5 --enableshadow
rootpw --iscrypted $1$1o751Xnc$kmQKHj6gtZ50IILNkHkkF0
selinux --permissive
reboot
firewall --enabled
skipx
key --skip

%packages
@ Base
device-mapper-multipath
ntp

%post
(
#set the time from a server, then use this to set the hwclock. Enable ntp
/usr/sbin/ntpdate 10.16.255.2
hwclock --systohc

# Configure iptables
/bin/cp /etc/sysconfig/iptables /tmp/iptables
/usr/bin/head -n -2  /tmp/iptables > /etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp --dport 54321 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp -m multiport --dports 5634:6166 -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp -m multiport --dport 16509 -j ACCEPT" >> /etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp --dport 49152:49216 -j ACCEPT" >> /etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -m physdev  --physdev-is-bridged -j ACCEPT" >> /etc/sysconfig/iptables
/usr/bin/tail -2  /tmp/iptables >> /etc/sysconfig/iptables

#disable LRO - BZ 518531
echo "options bnx2x disable_tpa=1" >> /etc/modprobe.conf

# turn on rhn osad
/sbin/chkconfig osad on

#get the latest software
/usr/bin/yum -y update

# fix issues with networks started and crashing the driver on the reboot
/bin/cp /etc/rc.d/rc.local /etc/rc.d/rc.local.shipped
cat <<'EOF'>>/etc/rc.d/rc.local
#Even though we didn't want the networks to have ONBOOT=yes, they do, so make sure all except eth0 do not
sed -i 's/ONBOOT=yes/ONBOOT=no/' /etc/sysconfig/network-scripts/ifcfg-eth[1-7]

#BZ602402
sed -i "s/modprobe -q bnx2i/#modprobe -q bnx2i/" /etc/init.d/iscsid
chkconfig iscsid off
# put standard rc.local back into place
/bin/mv /etc/rc.d/rc.local.shipped /etc/rc.d/rc.local

#reboot to get networks going (restart of network would not address crashed driver)
reboot
EOF

) >> /root/ks-post.log 2>&1
```

6.7 Configuring RHEV

Section **6.4.3** installed the Windows 2008 VM image used for the RHEV Manager. When accessing the RHEV-M VM for the first time after *sysprep* was run, the user must:

- set the region, time, and keyboard settings
- accept the license terms
- specify a password when logging in as Administrator
- optionally, adjust the screen resolution

Several other steps must be followed to complete the configuration such as:

- installing RHEV manager software
- configuring the RHEV environment
- uploading ISO images

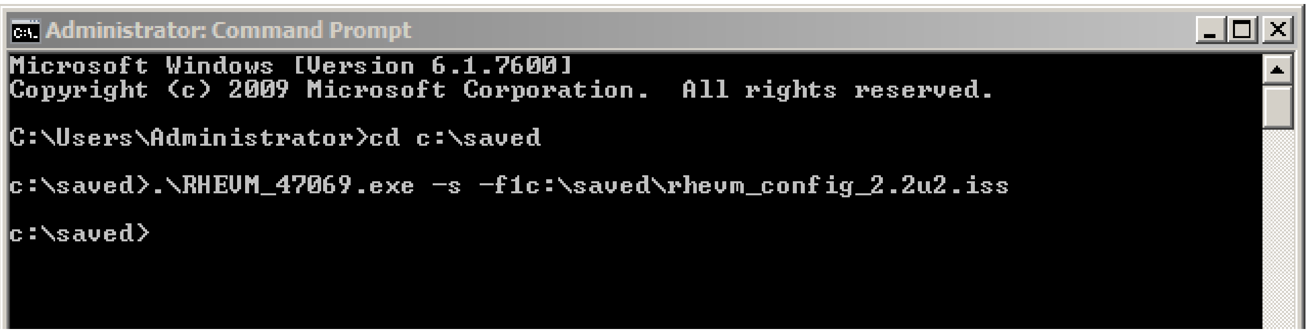
6.7.1 Installing RHEV Manager Software

The **Red Hat Cloud Foundations Edition One: Private IaaS Clouds** document demonstrated the user interface to installing RHEV-M. This could be followed here as well, however in the Windows image used for this project, an installation was recorded and the resulting answer file has been included in the *C:\saved* directory. A installation answer file can be created by specifying a “-r” when performing an installation. This generates the file *C:\windows\setup.iss* which may be renamed as desired and used when specifying the options “-s -f1c:\saved\rhevm_config_2.2u2.iss”.

Since this configuration is using local authentication, the domain name is the computer name. The computer name changes as part of the sysprep process, therefore it must be changed in the answer file prior to installing the RHEV Manager software. The current computer name can be obtain by viewing the computer's properties. Edit the rhevm_config_2.2u2.iss file using a preferred text editor and change the line containing “*Domain*=” to match that of the computer name.

Begin the install by opening a Command Prompt window, changing to the *C:\saved* directory, and issuing the command to install.

```
.\RHEVM_47069.exe -s -f1c:\saved\rhevm_config_2.2u2.iss
```



```
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Administrator>cd c:\saved

c:\saved>.\RHEVM_47069.exe -s -f1c:\saved\rhevm_config_2.2u2.iss

c:\saved>
```

6.7.2 Configure the RHEV Environment

Also included in the *C:\saved* directory of the Windows image used in this document is the following PowerShell script, which automates the following actions:

- log into the RHEV-M API
- create the Cloud-DC1 data center
- create the DC1-Clus1 cluster
- set the Cluster Policy to Even
- add first RHEL/KVM host
- create data storage
- create ISO library
- bring the data center online
- approve any RHEV-H hosts

RHEVMInitConfig.ps1

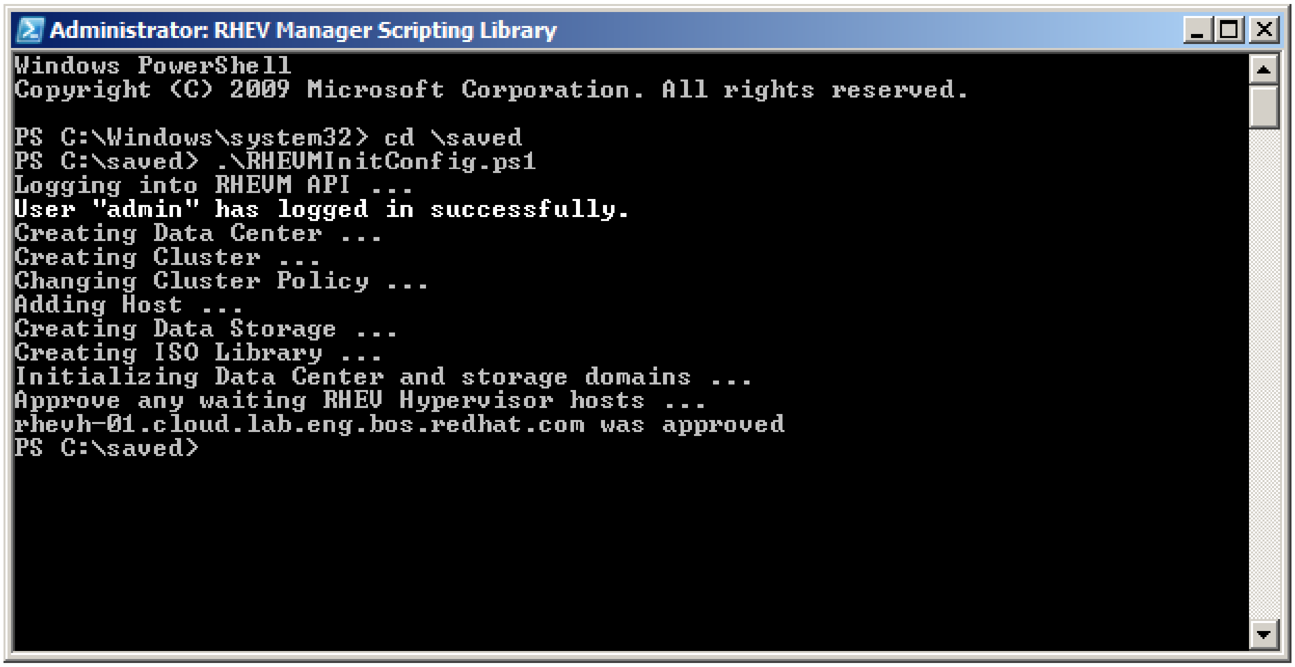
```
$cname = "$env:computername"

# Log in to RHEV-M API
write "Logging into RHEV-M API ..."
login-user admin 24^gold $cname

#create data center
write "Creating Data Center ..."
$dcversions = get-datacentercompatibilityversions
$dc = add-datacenter -Name "Cloud-DC1" -DataCenterType "FCP" -CompatibilityVersion
$dcversions[$dcversions.length-1]

#create cluster
write "Creating Cluster ..."
$clusversions = get-clustercompatibilityversions
$clus = add-cluster -ClusterName "DC1-Clus1" -DataCenterId $dc.DataCenterId -ClusterCpuName "Intel
```

```
Xeon Core i7"  -CompatibilityVersion $clusversions[$clusversions.length-1]

#change cluster policy
write "Changing Cluster Policy ..."
$clus.SelectionAlgorithm="EvenlyDistribute"
$clus.HighUtilization=80
$clus.CpuOverCommitDuratinMinutes=2
$clus = update-cluster $clus

#add host
write "Adding Host ..."
$rhelhost = add-host -Name "rhelh-01.cloud.lab.eng.bos.redhat.com" -Address rhelh-
01.cloud.lab.eng.bos.redhat.com -RootPassword "24^gold" -HostClusterId  $clus.ClusterId
-ManagementHostName 10.16.136.235 -ManagementType ilo -ManagementUser Administrator
-ManagementPassword "24^goldA" -AllowManagement

#wait for host to be up
$timeout=90
do {
  --$timeout
  $stat = (select-host rhelh-01).Status
  start-sleep -s 10
} while ( $timeout -and $stat -ne "Up" )

if ( $timeout -eq 0) { throw 'HOSTTIMEOUT' }

#create data storage
write "Creating Data Storage ..."
$storDev = get-storagedevices -HostId $rhelhost.HostId
$storDom = add-storagedomain -LunIdList $storDev.Id -hostId $rhelhost.HostId -DomainType Data
-StorageType FCP -Name 'fc1-1tb'

#create ISO Library
write "Creating ISO Library ..."
$isoDom = add-storagedomain -Storage "rhev-nfs.cloud.lab.eng.bos.redhat.com:/rhev/ISO" -hostId
$rhelhost.HostId -DomainType ISO -StorageType NFS -Name 'ISO_Library'

#Initialize data center with storage domains
write "Initializing Data Center and storage domains ..."
[system.collections.arraylist]$storArray = new-object System.Collections.ArrayList
$res=$storArray.Add($storDom.StorageDomainId)
$res=$storArray.Add($isoDom.StorageDomainId)
$initDC = init-DataCenter -DataCenterId $dc.DataCenterId -StorageDomainIdList $storArray

#wait for data center to be up
$timeout=90
do {
  --$timeout
  $stat = (select-datacenter Cloud-DC1).Status
  start-sleep -s 10
} while ( $timeout -and $stat -ne "Up" )

if ( $timeout -eq 0) { throw 'DATACENTERTIMEOUT' }

#Approve any rhev hosts that are present
write "Approve any waiting RHEV Hypervisor hosts ..."
foreach ($candidate in select-host | ? {$_.Status -eq "Pending Approval"})  {
  if ($candidate.Name -like "rhevh-01*") {
     $candidate.PowerManagement.Enabled = $true
     $candidate.PowerManagement.Address = "10.16.136.234"
     $candidate.PowerManagement.Type = "ilo"
     $candidate.PowerManagement.UserName = "Administrator"
     $candidate.PowerManagement.Password = "24^goldA"
     $updateHost = Update-Host -HostObject $candidate
  }
  $appHost = Approve-Host -HostObject $candidate  -ClusterObject $clus
  write-host $appHost.name was approved
}
```

At the end of this script, any RHEV-H hosts awaiting approval are approved. If a RHEV-H host was installed prior to the installation of the RHEV Manager software, a reboot of the RHEV-H communicates with the RHEV-M and marks itself as pending approval.

The script is called from the *Start -> All Programs -> Red Hat -> RHEV Manager -> RHEV Manager Scripting Library*. In this PowerShell window, change the working directory to the *C:\saved* directory and invoke the script.

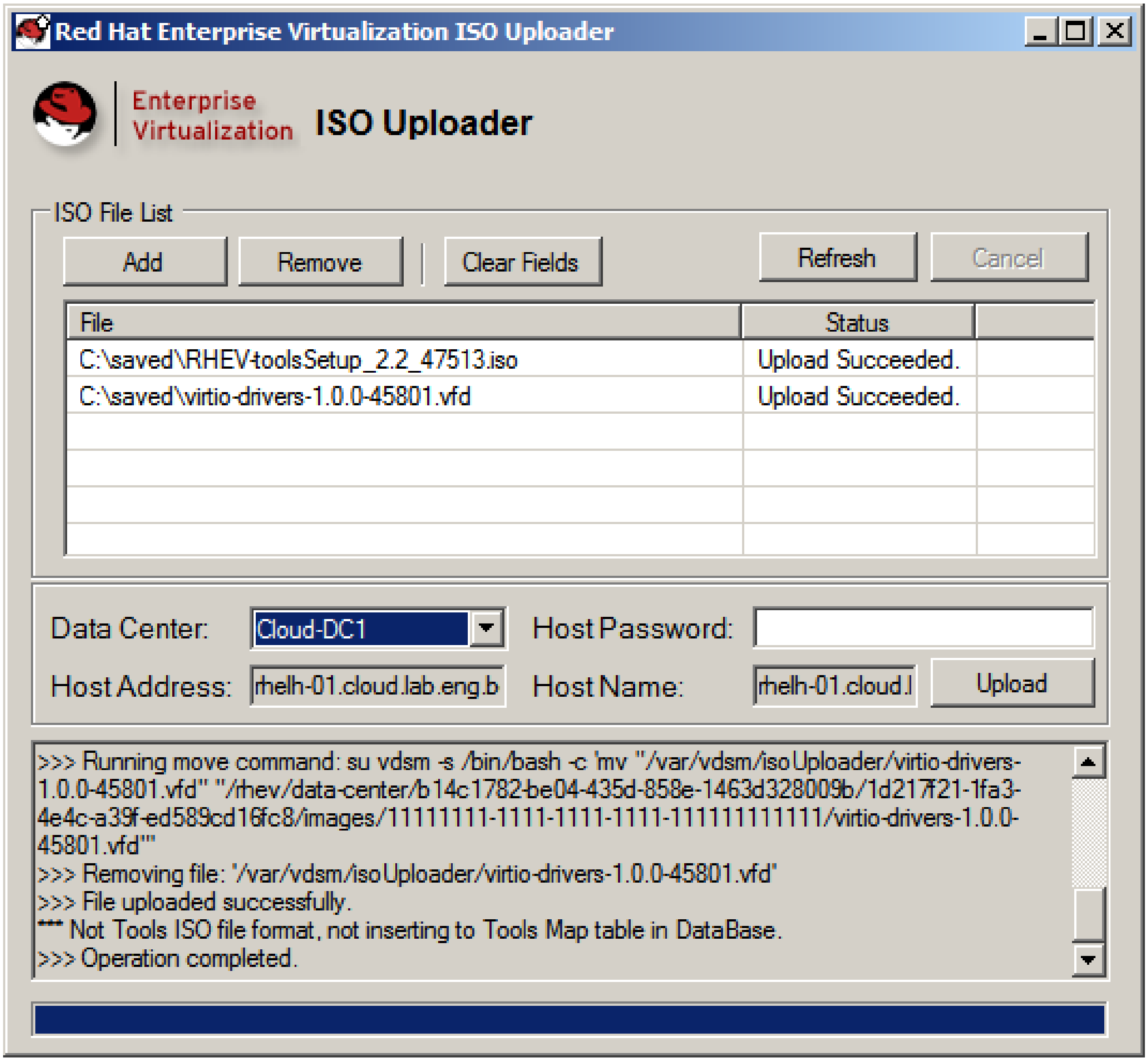
```
cd C:\saved
./RHEVMInitConfig.ps1
```



```
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> cd \saved
PS C:\saved> .\RHEVMInitConfig.ps1
Logging into RHEVM API ...
User "admin" has logged in successfully.
Creating Data Center ...
Creating Cluster ...
Changing Cluster Policy ...
Adding Host ...
Creating Data Storage ...
Creating ISO Library ...
Initializing Data Center and storage domains ...
Approve any waiting RHEV Hypervisor hosts ...
rhevh-01.cloud.lab.eng.bos.redhat.com was approved
PS C:\saved>
```

6.7.3 Upload ISO Images

With RHEV-M now operational, upload the guest tools ISO image and the virtio drivers virtual floppy disk. *Start -> All Programs -> Red Hat -> RHEV Manager -> ISO Uploader*.


Red Hat Enterprise Virtualization ISO Uploader
Enterprise Virtualization
ISO Uploader
ISO File List
Add
Remove
Clear Fields
Refresh
Cancel
Upload

File	Status
C:\saved\RHEV-toolsSetup_2.2_47513.iso	Upload Succeeded.
C:\saved\virtio-drivers-1.0.0-45801.vfd	Upload Succeeded.

Data Center: Cloud-DC1

Host Password:

Host Address: rhelh-01.cloud.lab.eng.b

Host Name: rhelh-01.cloud.l

```
>>> Running move command: su vdsm -s /bin/bash -c 'mv "/var/vdsm/isoUploader/virtio-drivers-
1.0.0-45801.vfd" "/rhev/data-center/b14c1782-be04-435d-858e-1463d328009b/1d217f21-1fa3-
4e4c-a39f-ed589cd16fc8/images/11111111-1111-1111-1111-111111111111/virtio-drivers-1.0.0-
45801.vfd"'
>>> Removing file: '/var/vdsm/isoUploader/virtio-drivers-1.0.0-45801.vfd'
>>> File uploaded successfully.
*** Not Tools ISO file format, not inserting to Tools Map table in DataBase.
>>> Operation completed.
```

7 Dynamic Addition and Removal of Hosts

As workloads ramp up and the demand for CPU cycles increases, additional hosts may be added to bear the burden of additional VMs that can provide added compute power. The scripts *instRHELH.sh* and *instRHEVH.sh* may be used to create a new host of each type.

7.1 RHEL / KVM Host Addition

The *instRHELH.sh* script requires a single passed parameter, the server blade profile name assigned in the Virtual Connect interface.

```
./instRHELH.sh rhelh-02
```

The script:

- creates the cobbler system entry
- presents storage to the host
- adds the new NFS export stanza to the cluster configuration file
- sets the boot order to boot PXE first
- registers the host with satellite after install
- sets the boot order to boot PXE last

and requires the user to add the newly created RHEL host in RHEV-M.

instRHELH.sh

```
#!/bin/bash
# This script will install and prepare a system to be a RHEL host

# Source env vars
if [[ -x varDefs.sh ]] ; then
  source varDefs.sh
elif [[ -x /root/varDefs.sh ]] ; then
  source /root/varDefs.sh
elif [[ -x /root/resources/varDefs.sh ]] ; then
  source /root/resources/varDefs.sh
elif [[ -x /root/distro/resources/varDefs.sh ]] ; then
  source /root/distro/resources/varDefs.sh
else
  echo "didn't find a varDefs.sh file"
fi

# The blade profile must be passed
if [[ $# -ne 1 ]]
then
  echo 'Usage - $0 <HP Virtual Connect profile name>'
  exit -1
else
  pname=$1
fi
```

```
vcmcommand --vcmurl //${LOGIN}:${VCM_PW}@${VCM_IP} show server | grep ${pname} > /dev/null 2>/dev/null
if [[ $? -ne 0 ]]
then
  echo "HP Virtual Connect profile $pname not found!"
  exit -2
fi

nblade=`vcmcommand --vcmurl //${LOGIN}:${VCM_PW}@${VCM_IP} show server | grep ${pname} | awk '{print $3}'`

iloIP=`oacommand --oaurl //${LOGIN}:${OA_PW}@${OA_IP} show server info ${nblade} | grep "IP Address" | awk '{print $3}'`

rawMAC=`vcmcommand --vcmurl //${LOGIN}:${VCM_PW}@${VCM_IP} show profile ${pname} | grep public | awk '{print $5}'`

#implement a semaphore to verify name and IP are unique
while [[ -e /tmp/AvailNameIP ]] ; do sleep 1; done
touch /tmp/AvailNameIP

IPname=`/root/resources/GetAvailRhelh.sh | awk '{print $1}'`

IPnum=`/root/resources/GetAvailRhelh.sh | awk '{print $2}'`

# Create cobbler system entry
echo -e "\nCreating cobbler system entry ...\n"
cobbler system add --name=${IPname} --profile=${RHELH_PROFILE} --mac=${rawMAC//-/:} --ip=${IPnum} --hostname=${IPname} --dns-name=${IPname} --kopts="console=ttyS0,115200 nostorage"
cobbler sync

#Remove semaphore
/bin/rm /tmp/AvailNameIP

# Present storage to hosts
echo -e "\nPresenting storage to host ${pname} ...\n"
/root/resources/prep_stor_host.sh ${pname}

# Update cluster configuration for NFS presentation

# create a semaphore for unique client names
while [[ -e /tmp/UpdateNFSClient ]] ; do sleep 1; done
touch /tmp/UpdateNFSClient

#Get next available client name
nfsClient=`/root/resources/GetAvailNFSClient.sh`

# add the export to the cluster configuration
addnfsexport -v -H ${MGMT1_IP} rhev-nfs-fs ${nfsClient} ${IPname} rw 1
```

```
# release semaphore
/bin/rm /tmp/UpdateNFSClient

# Get the count of systems registered with this name (should be 0)
initReg=`/root/resources/listRegSystems_infra.py | grep -c ${IPname}`
echo -e "\nNumber of systems registered with this name: ${initReg} ...\n"

# Set to boot PXE first
echo -e "\nChanging system boot order to boot network (PXE) first ...\n"
while [[ ! `ilocommand -i //${LOGIN}:${ILO_PW}@${iloIP} set /system1/bootconfig1/bootsource5
bootorder=1 | grep status=0` ]]; do sleep 10; done

# Reset node (power on node in case node was off)
echo -e "\nResetting server blade (power on in case blade was off) ...\n"
ilocommand -i //${LOGIN}:${ILO_PW}@${iloIP} power reset
ilocommand -i //${LOGIN}:${ILO_PW}@${iloIP} power on

# Wait for system to register with satellite indicating installation completion
echo -e "\nWaiting for system to register with satellite ...\n"
while [[ $initReg -ge `/root/resources/listRegSystems_infra.py | grep -c ${IPname}` ]]; do sleep 15; done
echo -e "\nSatellite registration complete ...\n"

# Set to boot PXE last
echo -e "\nChanging system boot order to boot network last ...\n"
while [[ ! `ilocommand -i //${LOGIN}:${ILO_PW}@${iloIP} set /system1/bootconfig1/bootsource5
bootorder=5 | grep status=0` ]]; do sleep 2; done
```

7.2 RHEV Host Addition

The *instRHEVH.sh* script installs and prepares a system for use as a RHEV host. It requires a single passed parameter, the server blade profile name assigned in the HP Virtual Connect interface, and requires the user to approve the newly created host in RHEV-M.

```
./instRHEVH.sh rhevh-02
```

instRHEVH.sh

```
#!/bin/bash
#
# This script will install and prepare a system for use as a RHEV host

# source env vars
if [[ -x varDefs.sh ]] ; then
  source varDefs.sh
elif [[ -x /root/varDefs.sh ]] ; then
  source /root/varDefs.sh
elif [[ -x /root/resources/varDefs.sh ]] ; then
  source /root/resources/varDefs.sh
elif [[ -x /root/distro/resources/varDefs.sh ]] ; then
  source /root/distro/resources/varDefs.sh
else
```

```
  echo "Didn't find a varDefs.sh file!"
fi

# The blade profile must be passed
if [[ $# -ne 1 ]]
then
  echo 'Usage - $0 <HP Virtual Connect profile name>'
  exit -1
else
  pname=$1
fi

# Implement a semaphore to verify that name and IP are unique
while [[ -e /tmp/AvailNameIP ]] ; do sleep 1; done
touch /tmp/AvailNameIP

vcmcommand --vcmurl //${LOGIN}:${VCM_PW}@${VCM_IP} show server | grep ${pname} > /dev/null 2>/dev/null
if [[ $? -ne 0 ]]
then
  echo "HP Virtual Connect profile $pname not found!"
  exit -2
fi

nblade=`vcmcommand --vcmurl //${LOGIN}:${VCM_PW}@${VCM_IP} show server | grep ${pname} | awk '{print $3}'`
iloIP=`oacommand --oaurl //${LOGIN}:${OA_PW}@${OA_IP} show server info ${nblade} | grep "IP Address" | awk '{print $3}'`
rawMAC=`vcmcommand --vcmurl //${LOGIN}:${VCM_PW}@${VCM_IP} show profile ${pname} | grep public | awk '{print $5}'`
IPname=`/root/resources/GetAvailRhevh.sh | awk '{print $1}'`
IPnum=`/root/resources/GetAvailRhevh.sh | awk '{print $2}'`

# Delete any previously defined cobbler system entry
if [[ `cobbler system list | grep ${IPname}` ]]
then
  cobbler system delete --name=${IPname}
fi

# Create cobbler system entry
echo -e "\nCreating cobbler system entry ...\n"
cobbler system add --name=${IPname} --profile=${RHEVH_PROFILE} --mac=${rawMAC//-/:} --ip=${IPnum} --hostname=${IPname} --dns-name=${IPname} --kopts="console=ttyS0,115200 nostorage"
cobbler sync

# Remove semaphore
/bin/rm /tmp/AvailNameIP

# Present storage to host
echo -e "\nPresenting storage to host ${pname} ...\n"
```

```
/root/resources/prep_stor_host.sh ${pname}

# Update cluster configuration for NFS presentation

# create a semaphore for unique client names
while [[ -e /tmp/UpdateNFSClient ]] ; do sleep 1; done
touch /tmp/UpdateNFSClient

#Get next available client name
nfsClient=`/root/resources/GetAvailNFSClient.sh`

# add the export to the cluster configuration
addnfsexport -v -H ${MGMT1_IP} rhev-nfs-fs ${nfsClient} ${IPname} rw 1

# release semaphore
/bin/rm /tmp/UpdateNFSClient

# Get the count of systems registered with this name (should be 0)
initReg=`/root/resources/listRegSystems_infra.py | grep -c ${IPname}`
echo -e "\nNumber of systems registered with this name: ${initReg} ...\n"

# Change system boot order to boot network (PXE) first
echo -e "\nChanging system boot order to boot network (PXE) first ...\n"
while [[ ! `ilocommand -i //${LOGIN}:${ILO_PW}@${iloIP} set /system1/bootconfig1/bootsource5
bootorder=1 | grep status=0` ]]; do sleep 10; done

# Reset server blade (power on in case blade was off)
echo -e "\nResetting server blade (power on in case blade was off) ...\n"
ilocommand -i //${LOGIN}:${ILO_PW}@${iloIP} power reset
ilocommand -i //${LOGIN}:${ILO_PW}@${iloIP} power on

# Wait for system to register with satellite indicating installation completion
echo -e "\nWaiting for system to register with satellite ...\n"
while [[ $initReg -ge `/root/resources/listRegSystems_infra.py | grep -c ${IPname}` ]]; do sleep 5; done
echo -e "\nSatellite registration complete ...\n"

# Change system boot order to boot network last
echo -e "\nChanging system boot order to boot network last ...\n"
while [[ ! `ilocommand -i //${LOGIN}:${ILO_PW}@${iloIP} set /system1/bootconfig1/bootsource5
bootorder=5 | grep status=0` ]]; do sleep 2; done
```

7.3 Host Removal

Subsequently, *remHost.sh* is used to remove a host of either type from the RHEV-M configuration.

```
./remHost.sh rhevh-02
```

remHost.sh

```
#!/bin/bash
# This script requires a host (hypervisor) name (also the profile name) as a passed parameter.
# It will remove a host from the cluster configuration, cobbler, and storage.
# It assumes that the host targeted for removal has been removed from RHEV-M.

# Source env vars
if [[ -x varDefs.sh ]] ; then
  source varDefs.sh
elif [[ -x /root/varDefs.sh ]] ; then
  source /root/varDefs.sh
elif [[ -x /root/resources/varDefs.sh ]] ; then
  source /root/resources/varDefs.sh
elif [[ -x /root/distro/resources/varDefs.sh ]] ; then
  source /root/distro/resources/varDefs.sh
else
  echo "didn't find a varDefs.sh file"
fi

# The host name must be passed
if [[ $# -ne 1 ]]
then
  echo 'Usage - $0 <host name for removal>'
  exit -1
else
  host=$1
fi

HOSTS=`host -l ${FQD} ${SAT_FQDN} |grep $host | cut -f1 -d'.' | sort`
if [[ ! ${HOSTS} ]]
then
  echo "No entry located for "$host
  exit -2
fi

# Remove cobbler system entry
echo -e "\nRemoving cobbler system entry ...\n"
hostFQDN=`cobbler system list |grep ${host}`
cobbler system delete --name=${hostFQDN}
cobbler sync

# Remove the NFS export resource from the cluster configuration
echo -e "\nRemoving cluster configuration entries ...\n"
```

```
nfsClient=`riccicmd -H ${MGMT1_IP} cluster configuration  | grep ${host} |cut -d\" -f4`
if [[ $? -ne 0 ]]
then
  echo "Cluster resource not found for ${host}!"
  exit -3
fi
delnfsexport --ricciroot=/.ricci  -H ${MGMT1_IP} rhev-nfs-fs ${nfsClient}

# Unpresent storage to removed host and delete host HBAs
echo -e "\nUnpresenting storage to removed host ${hostFQDN} ...\n"
/root/resources/rem_stor_host.sh ${host}

# Remove Satellite registration
/root/resources/wipeSatReg.py ${hostFQDN}

nblade=`vcmcommand --vcmurl //${LOGIN}:${VCM_PW}@${VCM_IP} show server | grep ${host} | awk
'{print $3}'`

iloIP=`oacommand --oaurl //${LOGIN}:${OA_PW}@${OA_IP} show server info ${nblade} | grep "IP
Address" | awk '{print $3}'`

ilocommand -i //${LOGIN}:${ILO_PW}@${iloIP} power off
```

- *rem_stor_host.sh* is called from within *remHost.sh* and unmaps any storage volumes previously mapped to the host

  rem_stor_host.sh

  ```
  #!/bin/bash
  # This script requires a host (hypervisor) name as a passed parameter.
  # It will unmap any volumes previously mapped to the host

  #source variables
  if [[ -x varDefs.sh ]]
  then
    source varDefs.sh
  elif [[ -x /root/varDefs.sh ]]
  then
    source /root/varDefs.sh
  elif [[ -x /root/resources/varDefs.sh ]]
  then
    source /root/resources/varDefs.sh
  elif [[ -x /root/distro/resources/varDefs.sh ]]
  then
    source /root/distro/resources/varDefs.sh
  else
    echo "didn't find a varDefs.sh file"
  fi

  # Confirm that a host name was passed
  if [[ $# -ne 1 ]]
  then
  ```

```
   echo "Usage: $0 <host for removal>\n"
   exit -1
else
   host=$1
fi

# Confirm that the parameter passed is an existing host
if [[ ! `sacommand --saurl //${MSA_USER}:${MSA_PW}@${MSA_IP} show hosts | grep ${host}` ]]
then
  echo ${host} " is not an existing host!"
  exit -2
fi

# Unmap any volumes that may be mapped to the specified host
for hostAlias in `sacommand --saurl //${MSA_USER}:${MSA_PW}@${MSA_IP} show hosts |grep $
{host} | awk '{print $2}'`
do
  for mappedVol in `sacommand --saurl //${MSA_USER}:${MSA_PW}@${MSA_IP} show host-
maps ${hostAlias} | head -5 | tail -1 | awk '{print $1}'`
  do
    sacommand --saurl //${MSA_USER}:${MSA_PW}@${MSA_IP} unmap volume host ${hostAlias}
${mappedVol}
  done
# Remove the HBAs of this host from the MSA storage array
#  sacommand --saurl //${MSA_USER}:${MSA_PW}@${MSA_IP} delete host ${hostAlias}
done
```

8 Creating VMs

This section includes the kickstart and other scripts used in the creation of these RHEV VMs:

- Basic RHEL
- RHEL with Java Application
- RHEL with JBoss Application
- RHEL MRG Grid Execute Nodes
- RHEL MRG Grid Rendering

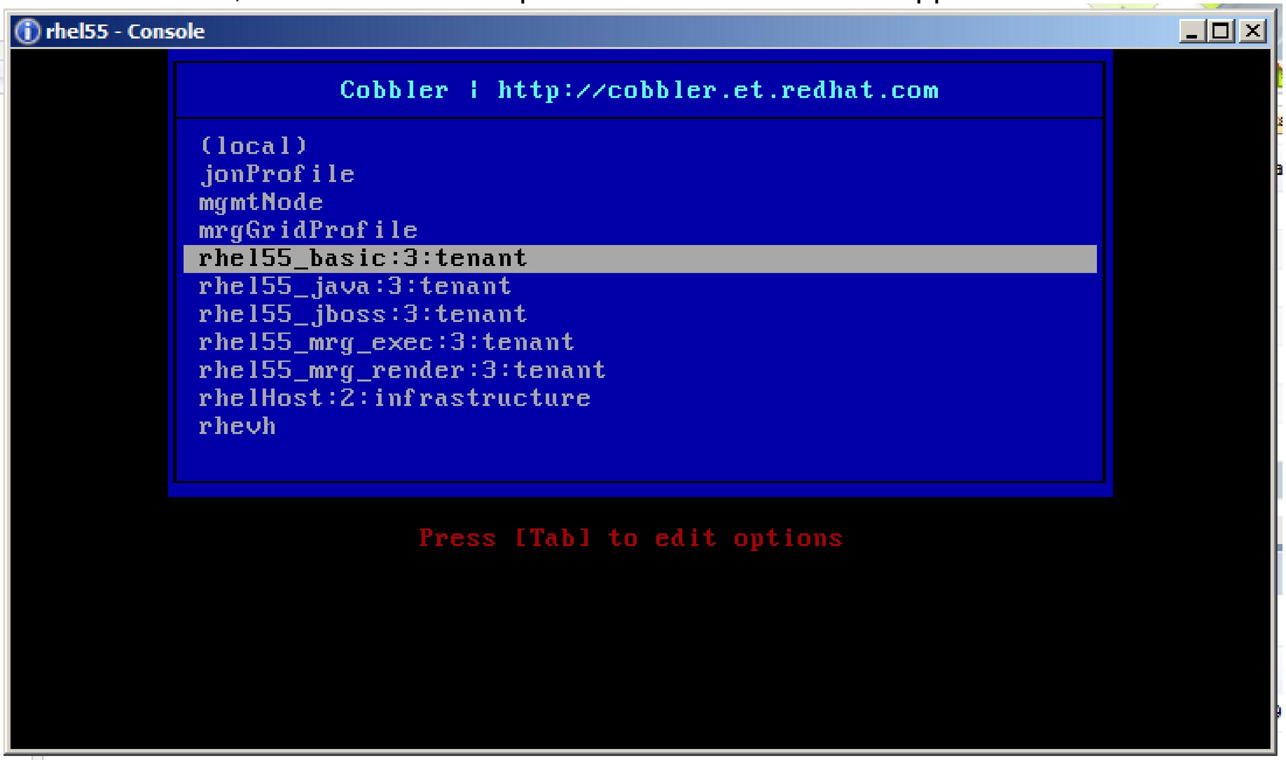
Each of the VMs are created following a similar procedure:

1. Log in to the RHEV Manager
2. Select the *Virtual Machines* tab
3. Select the *New Server* button
4. In the *New Server Virtual Machine* window
   a) In the *General* tab, provide the at least following:
      - Name: (e.g., *rhel55*)
      - Memory Size: (e.g., *1024 MB*)
      - Total Cores: (e.g., *2*)
      - Operation System: (e.g., *Red Hat Enterprise Linux 5.x x64*)

   b) In the *Boot Sequence* tab
      - Second Device: *Network (PXE)*

   c) Select *OK*
5. In the *New Virtual Machine – Guide Me* window, select *Configure Network Interfaces*
   a) Accept the defaults by selecting *OK*
6. In the *New Virtual Machine – Guide Me* window, select *Configure Virtual Disks*
   a) Specify Size (GB): (e.g., *6*)
   b) Click the *OK* button
7. In the *New Virtual Machine – Guide Me* window, select *Configure Later*
8. After the VM indicates a down status, select the VM and click the *Run* button
9. Click the *Console* button when selectable

10. In console,select the desired option when the PXE menu appears



```
Cobbler | http://cobbler.et.redhat.com

(local)
jonProfile
mgmtNode
mrgGridProfile
rhel55_basic:3:tenant
rhel55_java:3:tenant
rhel55_jboss:3:tenant
rhel55_mrg_exec:3:tenant
rhel55_mrg_render:3:tenant
rhelHost:2:infrastructure
rhevh

Press [Tab] to edit options
```

11. The VM installs and registers with the local RHN Satellite server

8.1 RHEL

The basic RHEL VM was created with following kickstart file, used to install RHEL 5.5 with the latest available updates from RHN.

rhel55_basic.ks

```
text
network --bootproto dhcp
lang en_US
keyboard us
url --url http://sat-vm.cloud.lab.eng.bos.redhat.com/ks/dist/ks-rhel-x86_64-server-5-u5
zerombr
clearpart --linux
part /boot --fstype=ext3 --size=200
part pv.01 --size=1000 --grow
part swap --size=2000   --maxsize=20000
volgroup rhel55 pv.01
logvol / --vgname=rhel55 --name=rootvol --size=1000 --grow
bootloader --location mbr
timezone America/New_York
auth --enablemd5 --enableshadow
rootpw --iscrypted $1$1o751Xnc$kmQKHj6gtZ50IILNkHkkF0
```

```
selinux --permissive
reboot
firewall --enabled
skipx
key --skip

%packages
@ Base

%post
(
/usr/bin/yum -y update
) >> /root/ks-post.log 2>&1
```

8.2 RHEL with Java Application

A self-starting Java workload is demonstrated with VMs created using the following kickstart to:

- install RHEL 5.5 with OpenJDK
- install the latest OS updates
- execute any queued actions on RHN to sync with the activation key

rhel55_java.ks

```
install
text
network --bootproto dhcp
lang en_US
keyboard us
url --url http://sat-vm.cloud.lab.eng.bos.redhat.com/ks/dist/ks-rhel-x86_64-server-5-u5
zerombr
clearpart --linux
part /boot --fstype=ext3 --size=200
part pv.01 --size=1000 --grow
part swap --size=2000   --maxsize=20000
volgroup rhel55 pv.01
logvol / --vgname=rhel55 --name=rootvol --size=1000 --grow
bootloader --location mbr
timezone America/New_York
auth --enablemd5 --enableshadow
rootpw --iscrypted $1$1o751Xnc$kmQKHj6gtZ50IILNkHkkF0
selinux --permissive
reboot
firewall --enabled
skipx
key --skip

%packages
@ Base
java-1.6.0-openjdk
```

```
%post
(
# execute any queued actions on RHN to sync with the activation key
rhn_check -vv

/usr/bin/yum -y update

) >> /root/ks-post.log 2>&1
```

8.3 RHEL with JBoss

A VM demonstrating JBoss EAP functionality is created using the following kickstart to:

- install RHEL 5.5 with OpenJDK
- set required firewall ports
- download, install and configure both JBoss EAP and the JON agent
- deploy the Java test application
- configure JBoss and the JON agent to auto start at VM boot
- install the latest OS updates

rhel55_jboss.ks

```
install
text
network --bootproto dhcp
lang en_US
keyboard us
url --url http://sat-vm.cloud.lab.eng.bos.redhat.com/ks/dist/ks-rhel-x86_64-server-5-u5
zerombr
clearpart --linux
part /boot --fstype=ext3 --size=200
part pv.01 --size=1000 --grow
part swap --size=2000   --maxsize=20000
volgroup rhel55 pv.01
logvol / --vgname=rhel55 --name=rootvol --size=1000 --grow
bootloader --location mbr
timezone America/New_York
auth --enablemd5 --enableshadow
rootpw --iscrypted $1$1o751Xnc$kmQKHj6gtZ50IILNkHkkF0
selinux --permissive
reboot
firewall --enabled
skipx
key --skip

%packages
@ Base
java-1.6.0-openjdk
```

```
%post
(
# set required firewall ports
/bin/cp /etc/sysconfig/iptables /tmp/iptables
/usr/bin/head -n -2  /tmp/iptables > /etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp --dport 22 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp --dport 1098 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp --dport 1099 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp --dport 3873 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp --dport 4444 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp --dport 4445 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp --dport 4446 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp --dport 4457 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp --dport 8009 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp --dport 8080 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp --dport 8083 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp --dport 1100 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp --dport 1101 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp --dport 1102 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp --dport 1161 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp --dport 1162 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp --dport 3528 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp --dport 4447 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp --dport 7900 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p udp --dport 1102 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p udp --dport 1161 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p udp --dport 1162 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp --dport 3528 -m state --state NEW -j ACCEPT" >>
```

```
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp --dport 4447 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp --dport 7900 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p udp --dport 43333 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p udp --dport 45551 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p udp --dport 45556 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p udp --dport 45557 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p udp --dport 45668 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p udp --dport 45557 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p udp --dport 5432 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp --dport 67 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp --dport 68 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/bin/echo "-A RH-Firewall-1-INPUT -p tcp --dport 16163 -m state --state NEW -j ACCEPT" >>
/etc/sysconfig/iptables
/usr/bin/tail -2  /tmp/iptables >> /etc/sysconfig/iptables

# download, install and configure JBoss EAP
cd /root
wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/jboss-eap-default.GA.zip
unzip jboss-eap-*.GA.zip
cd jboss-eap*/jboss-as/server/default/conf/props
sed -i 's/# admin=admin/admin=24^gold/' jmx-console-users.properties

# download, install and configure the JON agent
cd /root
wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/rhq-enterprise-agent-default.GA.jar
java -jar /root/rhq-enterprise-agent-default.GA.jar --install
cd /root/rhq-agent/conf
line=`grep -n "key=\"rhq.agent.configuration-setup-flag" agent-configuration.xml | cut -d: -f1`
before=`expr $line - 1`
after=`expr $line + 1`
sed -i -e "${after}d" -e "${before}d" agent-configuration.xml
sed -i -e '/rhq.agent.configuration-setup-flag/s/false/true/g' agent-configuration.xml
line=`grep -n "key=\"rhq.agent.server.bind-address" agent-configuration.xml | cut -d: -f1`
before=`expr $line - 1`
after=`expr $line + 1`
sed -i -e "${after}d" -e "${before}d" agent-configuration.xml
sed -i -e "/rhq.agent.server.bind-address/s/value=\".*\"/value=\"jon-vm.cloud.lab.eng.bos.redhat.com\"/g"
agent-configuration.xml
```

```
cd /root/rhq-agent/bin
\mv rhq-agent-env.sh rhq-agent-env.sh.orig
wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/rhq-agent-env.sh

# deploy the test app
cd /root/jboss-eap*/jboss-as/server/default/deploy
wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/jboss-seam-booking-ds.xml
wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/jboss-seam-booking.ear

# configure JBoss and JON agent to auto start
cd /etc/init.d
wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/jboss-eap
sed -i -e "s/readlink/readlink -e/g" /root/rhq-agent/bin/rhq-agent-wrapper.sh
ln -s /root/rhq-agent/bin/rhq-agent-wrapper.sh .
chmod +x jboss-eap rhq-agent-wrapper.sh
/sbin/chkconfig --add jboss-eap
/sbin/chkconfig jboss-eap on
/sbin/chkconfig --add rhq-agent-wrapper.sh
/sbin/chkconfig rhq-agent-wrapper.sh on

#update software
/usr/bin/yum -y update

) >> /root/ks-post.log 2>&1
```

1. Obtained by the kickstart, the *rhq-agent-env.sh* environment configuration file is customized to define the following variables according to the working environment:
    - RHQ_AGENT_HOME
    - RHQ_AGENT_JAVA_HOME
    - RHQ_AGENT_CMDLINE_OPTS
    - RHQ_AGENT_PIDFILE_DIR

    The configuration file used is located in Appendix **A.5.3**

8.4 RHEL MRG Grid Execute Nodes

VMs for use as MRG Grid execute nodes are created using the following kickstart to:

- install RHEL 5.5 with OpenJDK
- set required firewall ports
- execute any queued actions on RHN to sync with the activation key
- download the MRG configuration files
- install the latest OS updates
- configure MRG, NTP, the sesame database manager, and the AMQP broker daemon to auto start at VM boot

rhel55_mrg_exec.ks

```
install
```

```
text
network --bootproto dhcp
lang en_US
keyboard us
url --url http://sat-vm.cloud.lab.eng.bos.redhat.com/ks/dist/ks-rhel-x86_64-server-5-u5
zerombr
clearpart --linux
part /boot --fstype=ext3 --size=200
part pv.01 --size=1000 --grow
part swap --size=2000   --maxsize=20000
volgroup MRGGRID pv.01
logvol / --vgname=MRGGRID --name=rootvol --size=1000 --grow
bootloader --location mbr
timezone America/New_York
auth --enablemd5 --enableshadow
rootpw --iscrypted $1$1o751Xnc$kmQKHj6gtZ50IILNkHkkF0
selinux --permissive
reboot
firewall --enabled
skipx
key --skip

%packages
@ Base

%post
(
/bin/cp /etc/sysconfig/iptables /tmp/iptables
/usr/bin/head -n -2  /tmp/iptables > /etc/sysconfig/iptables
cat <<EOF>>/etc/sysconfig/iptables
-A RH-Firewall-1-INPUT -p tcp --dport 4672 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 4672 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 5672 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 5672 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 45672 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 45672 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 9618 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 9618 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 9614 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 9614 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 9600:9700 -m state --state ESTABLISHED,NEW -j
ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 9600:9700 -m state --state ESTABLISHED,NEW -j
ACCEPT
EOF
/usr/bin/tail -2  /tmp/iptables >> /etc/sysconfig/iptables

# execute any queued actions on RHN to sync with the activation key
rhn_check -vv
#Get configuration files
wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/sesame.conf -O /etc/sesame/sesame.conf
wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/condor_config.local.mrgexec -O
/var/lib/condor/condor_config.local

/usr/bin/yum -y update

chkconfig condor on
chkconfig qpidd on
condor_status -any
chkconfig sesame on
chkconfig ntpd on

) >> /root/ks-post.log 2>&1
```

The kickstart installs a simple RHEL VM including the software and configuration as a MRG Grid Execute node. The following example creates the node, creates a template from it, instantiates several nodes from the template, and executes a MRG Grid job that searches for perfect numbers.

1. Following the procedure outlined in Section **8** above, create a VM using the *rhel55_mrg_exec:3:tenant* profile. Specify 1 CPU, 1 GB of memory, and 5 GB for a local disk size.
2. After the VM installation completes, log in and make the following customizations.
   a) Setup the host to register with RHN Satellite on the next boot
      - Determine the command that was used to register this VM

        ```
        grep rhnreg /root/cobbler.ks
        rhnreg_ks --serverUrl=https://sat-
        vm.cloud.lab.eng.bos.redhat.com/XMLRPC
        --sslCACert=/usr/share/rhn/RHN-ORG-TRUSTED-SSL-CERT
        --activationkey=3-d18b20de1c22dcdcbe845577b068ed76,3-
        tenantMRGGridExec
        ```

      - Save a copy of *rc.local*

        ```
        cp /etc/rc.d/rc.local /etc/rc.d/rc.localpre
        ```

      - Add commands to *rc.local* to register the system with satellite using the '--force ' option and move the copy back into place.

        ```
        echo “rhnreg_ks--force \
          --serverUrl= https://sat-vm.cloud.lab.eng.bos.redhat.com/XMLRPC\
          --sslCACert=/usr/share/rhn/RHN-ORG-TRUSTED-SSL-CERT \
          --activationkey=3-d18b20de1c22dcdcbe845577b068ed76, \
          3-tenantMRGGridExec” >> /etc/rc.d/rc.local
        echo “mv /etc/rc.d/rc.local.pre /etc/rc.d/rc.local” >> \
          /etc/rc.d/rc.local
        ```

   b) Remove the line that defines the hostname in */etc/sysconfig/network*

      ```
      cp /etc/sysconfig/network /tmp
      grep -v “HOSTNAME=” /tmp/network > /etc/sysconfig/network
      ```

3. Shut down the VM
4. In the RHEV-M *Virtual Machines* tab, select the VM that was just shut down and click the *Make Template* button specifying a *Name*.
5. Create several VMs from the template using the *add-vms.ps1* script

```
# add-vms
#   tempName  - source template (can not be Blank)
#   baseName - base name of created guest (default: guest)
#   num    - number to create (default: 1)
#   run   -start VMs (default: no)

Param($baseName = 'guest', $tempName, $num = 1, [switch]$run)

if ($tempName -eq $null) {
  write-host "Must specify a template!"
  exit
}

$my_clusId = -1;

$my_temp = select-template -SearchText $tempName
if ($my_temp -eq $null)
{
  Write-host "No matching templates found!"
  exit
} elseif ($my_temp.count -gt 1) {
  Write-host "Too many matching templates found!"
  exit
} elseif ($my_temp.name -eq "Blank") {
  Write-host "Can not use Blank template!"
  exit
}

#search for matching basenames
$matches =  select-vm -searchtext  "$baseName" | where {$_.name -like "$baseName*"}
if ($matches -ne $null) {
   $measure = $matches | select-object name |  foreach {  $_.name.Replace("$baseName","") } |
measure-object -max
   $start = $measure.maximum + 1
   $x = $matches | select-object -first 1
   $my_clusId = $x.HostClusterId
} else {
   $start = 1
}

$id = $my_temp.HostClusterId

$clus = select-cluster | where { $_.ClusterID -eq $id }
if ($clus -ne $null) {
   if ($clus.IsInitialized -eq $true) {
     $my_clusId = $id
   } else {
     write-host "Cluster of Template is not initialized!"
     exit
   }
}

#loop over adds async
for ($i=$start; $i -lt $start + $num; $i++) {
#  write-host "-name $baseName$i -templateobject $my_temp -HostClusterId $my_clusId
-copytemplate -Vmtype server"
  if ( $run -eq $true ) {
    $my_vm = add-vm -name $baseName$i -templateobject $my_temp -HostClusterId $my_clusId
-copytemplate -Vmtype server

    #Until BZ 617730 and 617725 are addressed, use this work around
    $my_vm.DisplayType = 'VNC'
    $uv = update-vm -vmobject $my_vm

    $sv = start-vm -VmObject $my_vm

  } else {
    $my_vm = add-vm -name $baseName$i -templateobject $my_temp -HostClusterId $my_clusId
-copytemplate -Vmtype server -Async

    #Until BZ 617730 and 617725 are addressed, use this work around
    $my_vm.DisplayType = 'VNC'
    $my_vm.DisplayType = 'VNC'

    $uv = update-vm -vmobject $my_vm
  }
}
```

6. Call the script from a RHEV Manager Scripting Library window

   ```
   cd \saved
   .\add-vms.ps1 -baseName exec -tempName mrgexec_template -num 5
   ```

7. Log into the MRG Manager node as 'admin'
8. Create 50 separate submit jobs to search a range up to 1,000,000

   ```
   cd perfect
   ./mk_jobs.sh 1000000 50
   ```

9. Submit the jobs

   ```
   ./submit_jobs.sh
   ```

8.5 RHEL MRG Grid Rendering

A VM to execute the MRG Grid rendering application is created using the following kickstart to:

- install RHEL 5.5 with the Simple Direct Media Layer (SDL) library
- set required firewall ports
- execute any queued actions on RHN to sync with the activation key
- download the MRG configuration file
- install and configure the rendering application
- install the command line video decoding and encoding tool (mencoder)
- install the 3D content creation suite (blender)
- install the latest OS updates
- configure MRG, NTP, the sesame database manager, and the AMQP broker daemon to auto start at VM boot

rhel55_mrg_render.ks

```
install
text
network --bootproto dhcp
lang en_US
keyboard us
url --url http://sat-vm.cloud.lab.eng.bos.redhat.com/ks/dist/ks-rhel-x86_64-server-5-u5
zerombr
clearpart --linux
part /boot --fstype=ext3 --size=200
part pv.01 --size=1000 --grow
part swap --size=2000   --maxsize=20000
volgroup MRGGRID pv.01
logvol / --vgname=MRGGRID --name=rootvol --size=1000 --grow
bootloader --location mbr
timezone America/New_York
auth --enablemd5 --enableshadow
rootpw --iscrypted $1$1o751Xnc$kmQKHj6gtZ50IILNkHkkF0
selinux --permissive
reboot
firewall --enabled
skipx
key --skip
user --name=admin --password=$1$1o751Xnc$kmQKHj6gtZ50IILNkHkkF0 --iscrypted

%packages

@ Base
SDL

%post
(
/bin/cp /etc/sysconfig/iptables /tmp/iptables
/usr/bin/head -n -2  /tmp/iptables > /etc/sysconfig/iptables
cat <<EOF>>/etc/sysconfig/iptables
-A RH-Firewall-1-INPUT -p tcp --dport 4672 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 4672 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 5672 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 5672 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 45672 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 45672 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 9618 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 9618 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 9614 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 9614 -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 9600:9700 -m state --state ESTABLISHED,NEW -j
ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 9600:9700 -m state --state ESTABLISHED,NEW -j
ACCEPT
EOF
/usr/bin/tail -2  /tmp/iptables >> /etc/sysconfig/iptables

# register system with satellite
rhn_check -vv

#Get configuration files
wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/condor_config.local.mrgexecrender -O
/var/lib/condor/condor_config.local

# setup render
mkdir /home/admin/render
chown admin:admin /home/admin/render
cat <<EOF>>/etc/fstab
mrg-vm.cloud.lab.eng.bos.redhat.com:/home/admin/render /home/admin/render nfs defaults 0 0
EOF

#install blender
wget http://sat-vm.cloud.lab.eng.bos.redhat.com/pub/resources/blender-2.48a-linux-glibc236-py24-
x86_64-static.tar.bz2 -O /tmp/blender-2.48a-linux-glibc236-py24-x86_64-static.tar.bz2
su - admin -c "bzcat /tmp/blender-2.48a-linux-glibc236-py24-x86_64-static.tar.bz2 | tar xvf -"

/usr/bin/yum -y update

chkconfig condor on
chkconfig qpidd on
condor_status -any
chkconfig sesame on
chkconfig ntpd on

# install mencoder
rpm -Uvh http://download.fedora.redhat.com/pub/epel/5/i386/epel-release-5-4.noarch.rpm
rpm -Uvh http://download1.rpmfusion.org/free/el/updates/testing/5/i386/rpmfusion-free-release-5-
```

```
0.1.noarch.rpm http://download1.rpmfusion.org/nonfree/el/updates/testing/5/i386/rpmfusion-nonfree-
release-5-0.1.noarch.rpm
wget http://irish.lab.bos.redhat.com/pub/kits/fribidi-0.10.7-5.1.i386.rpm
wget http://irish.lab.bos.redhat.com/pub/kits/fribidi-0.10.7-5.1.x86_64.rpm
yum -y localinstall fribidi-0.10.7-5.1.x86_64.rpm
yum -y localinstall fribidi-0.10.7-5.1.i386.rpm
yum -y install mencoder

) >> /root/ks-post.log 2>&1
```

The kickstart creates a RHEL VM that mounts the render directory from MRG Manger. Both MRG Grid and rendering software are installed on the node. The following example creates the node and run a MRG Grid job that renders a small clip from a larger movie.

1. Following the procedure outlined in Section **8**, create a VM using the *rhel55_mrg_render:3:tenant* profile. Specify 4 CPUs, 4 GB of memory, and a size of 8 GB for a local disk size. Creating multiple VMs using a template or via the full install process speeds up rendering.

2. Log into the MRG Manager node as 'admin'

3. Create the submit jobs to render the clip

   ```
   cd perfect
   ruby setup_jobs.rb
   ```

4. Submit the job

   ```
   cd production/jobs
   ./start_job.sh
   ```

9 References

1. The NIST Definition of Cloud Computing
   Version 15
   07 October 2009
   http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc

2. Above the Clouds: A Berkley View of Cloud Computing
   Technical Report No. UCB/EECS-2009-28
   Department of Electrical Engineering and Computer Science
   University of California at Berkeley
   http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf

3. Cloud Computing Use Cases White Paper
   (A white paper produced by the Cloud Computing Use Case Discussion Group)
   Version 2.0
   30 October 2009
   http://groups.google.com/group/cloud-computing-use-cases

4. Configuring and Managing a Red Hat Cluster
   http://www.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5.2/html/Cluster_Administration/index.html

5. Red Hat Cloud Foundations Reference Architecture
   Edition One: Private Iaas Clouds
   https://inquiries.redhat.com/go/redhat/cloud-foundations

6. Red Hat Cloud Foundations Reference Architecture
   Edition One: Hybrid Iaas Clouds
   https://inquiries.redhat.com/go/redhat/hybrid-iaas-clouds

7. Red Hat Enterprise Virtualization - API Guide
   http://www.redhat.com/docs/en-US/Red_Hat_Enterprise_Virtualization/Red_Hat_Enterprise_Virtualization_for_Servers/2.1/pdf/API_Guide/API_Guide.pdf

Appendix A: Configuration Files, Images, Etc.

This appendix contains various configuration files, images, tar/zip files used in the construction of the infrastructure or in the various use cases demonstrated.

A.1 Management Nodes

A.1.1 multipath.conf.template

The multipath.conf.template file provides the settings to multipath for the specific hardware in use in this configuration. It is called a template because the aliases for the WWIDs of volumes created are appended to create the device-mapper multipath configuration file */etc/multipath.conf*.

```
defaults {
        udev_dir                /dev
        polling_interval        5
        selector                "round-robin 0"
        path_grouping_policy    multibus
        getuid_callout          "/sbin/scsi_id -g -u -s /block/%n"
        prio_callout            /bin/true
        path_checker            readsector0
        rr_min_io               100
        max_fds                 8192
        rr_weight               priorities
        failback                immediate
        no_path_retry           5
        user_friendly_names     yes
}
devices {
        device {
                vendor                  "HP"
                product                 "MSA2[02]12fc|MSA2012i"
                getuid_callout          "/sbin/scsi_id -g -u -s /block/%n"
                hardware_handler        "0"
                path_selector           "round-robin 0"
                path_grouping_policy    multibus
                failback                immediate
                rr_weight               uniform
                no_path_retry           3
                rr_min_io               100
                path_checker            tur
        }
        device {
                vendor                  "HP"
                product                 "MSA2312fc|MSA2324fc"
                getuid_callout          "/sbin/scsi_id -g -u -s /block/%n"
                hardware_handler        "0"
                path_selector           "round-robin 0"
                prio_callout            "/sbin/mpath_prio_alua /dev/%n"
                path_grouping_policy    group_by_prio
                failback                immediate
                rr_weight               uniform
                no_path_retry           3
                rr_min_io               100
                path_checker            tur
        }
}
```

A.1.2 cluster.conf

A cluster is instantly formed by depositing the following */etc/cluster/cluster.conf* file in place and starting the cluster daemons. Additions to *cluster.conf* to add the NFS clients are handled by the `riccicmd` scripting library.

```
<?xml version="1.0"?>
<cluster alias="mgmt" config_version="1" name="mgmt">
        <fence_daemon clean_start="0" post_fail_delay="0" post_join_delay="30"/>
        <clusternodes>
                <clusternode name="mgmt1" nodeid="1" votes="1">
                        <fence>
                                <method name="1">
                                        <device name="ra-c7000-01-db1-ilo"/>
                                </method>
                        </fence>
                </clusternode>
                <clusternode name="mgmt2" nodeid="2" votes="1">
                        <fence>
                                <method name="1">
                                        <device name="ra-c7000-01-db2-ilo"/>
                                </method>
                        </fence>
                </clusternode>
        </clusternodes>
        <cman expected_votes="1" two_node="1" />
        <fencedevices>
                <fencedevice agent="fence_ilo" hostname="ra-c7000-01-db1-ilo.cloud.lab.eng.bos.redhat.com"
login="Administrator" name="ra-c7000-01-db1-ilo" passwd="24^goldA"/>
                <fencedevice agent="fence_ilo" hostname="ra-c7000-01-db2-ilo.cloud.lab.eng.bos.redhat.com"
login="Administrator" name="ra-c7000-01-db2-ilo" passwd="24^goldA"/>
        </fencedevices>
        <rm>
                <failoverdomains>
                        <failoverdomain name="mgmt_fod" nofailback="0" ordered="0" restricted="0">
                                <failoverdomainnode name="mgmt1" priority="1"/>
                                <failoverdomainnode name="mgmt2" priority="1"/>
                        </failoverdomain>
                </failoverdomains>
        <resources>
            <ip address="10.16.136.20" monitor_link="1"/>
            <nfsexport name="rhev-nfs-exp"/>
            <fs device="/dev/mapper/MgmtServicesVG-RHEVNFSvol" force_fsck="0" force_unmount="0" fsid="49243" fstype="ext3" mountpoint="/rhev" name="rhev-nfs-fs" options="rw" self_fence="0"/>
        </resources>
        <vm autostart="1" domain="mgmt_fod" exclusive="0" hypervisor="qemu" max_restarts="2" migrate="live" name="sat-vm" path="/gfs2_vol/vmconfig" recovery="restart" restart_expire_time="60"/>
        <service autostart="1" domain="mgmt_fod" exclusive="0" max_restarts="2" name="rhev-nfs" recovery="restart" restart_expire_time="60">
            <ip ref="10.16.136.20"/>
            <fs ref="rhev-nfs-fs">
                <nfsexport ref="rhev-nfs-exp">
                </nfsexport>
            </fs>
        </service>
        <vm autostart="1" domain="mgmt_fod" exclusive="0" hypervisor="qemu" max_restarts="2" migrate="live" name="rhevm-vm" path="/gfs2_vol/vmconfig" recovery="restart" restart_expire_time="60"/>
        <vm autostart="1" domain="mgmt_fod" exclusive="0" hypervisor="qemu" max_restarts="2" migrate="live" name="mrg-vm" path="/gfs2_vol/vmconfig" recovery="restart" restart_expire_time="60"/>
        <vm autostart="1" domain="mgmt_fod" exclusive="0" max_restarts="2" name="jon-vm" path="/gfs2_vol/vmconfig" recovery="restart" restart_expire_time="60"/>
    </rm>
    <totem token="55000" consensus="60000"/>
</cluster>
```

A.1.3 named.conf.slave

While the satellite system is the named master, each cluster node acts as a slave server. This file is renamed to */etc/named.conf*.

```
options {
        directory "/var/named";
        forwarders { 10.16.255.2 port 53; 10.16.255.3 port 53; };
};

zone "cloud.lab.eng.bos.redhat.com." { type slave; file "cloud.lab.eng.bos.redhat.com"; masters {10.16.136.1;}; };
zone "140.16.10.in-addr.arpa." { type slave; file "10.16.140"; masters {10.16.136.1;}; };
zone "141.16.10.in-addr.arpa." { type slave; file "10.16.141"; masters {10.16.136.1;}; };
zone "142.16.10.in-addr.arpa." { type slave; file "10.16.142"; masters {10.16.136.1;}; };
zone "143.16.10.in-addr.arpa." { type slave; file "10.16.143"; masters {10.16.136.1;}; };
zone "139.16.10.in-addr.arpa." { type slave; file "10.16.139"; masters {10.16.136.1;}; };
zone "138.16.10.in-addr.arpa." { type slave; file "10.16.138"; masters {10.16.136.1;}; };
zone "137.16.10.in-addr.arpa." { type slave; file "10.16.137"; masters {10.16.136.1;}; };
zone "136.16.10.in-addr.arpa." { type slave; file "10.16.136"; masters {10.16.136.1;}; };
```

A.2 Satellite

A.2.1 answers.txt

```
# Administrator's email address.  Required.
# Multiple email addresses can be used, separated with commas.
#
# Example:
# admin-email = user@example.com, otheruser@example.com

admin-email = <user>@<companyName>.com

## RHN connection information.
#
# Passed to rhn-register to register the system if it is not already
# registered.
#
# Only required if the system is not already registered, or if the
# '--re-register' commandline option is used.  Not used at all if the
# '--disconnected' commandline option is used.

rhn-username = <username>
rhn-password = <password>

# HTTP proxy.  Not Required.
#
# Example:
# rhn-http-proxy = proxy.example.com:8080

rhn-http-proxy =
rhn-http-proxy-username =
rhn-http-proxy-password =

# RHN Profile name.  Not required.  Defaults to the system's hostname
# or whatever 'hostname' is set to.

# rhn-profile-name =

## SSL certificate information.

# The name of your organization.  Required.
#
# Example:
# ssl-set-org = Riboflavin, Inc.

ssl-set-org = Red Hat

# The unit within the organization that the satellite is assigned to.
# Not required.
#
```

```
# Example:
# ssl-set-org-unit = Information Systems Department

ssl-set-org-unit = Reference Architecture

# Location information for the SSL certificates.  Required.
#
# Example:
# ssl-set-city = New York
# ssl-set-state = NY
# ssl-set-country = US

ssl-set-city = Westford
ssl-set-state = MA
ssl-set-country = US

# Password for CA certificate.  Required.  Do not lose or forget this
# password!
#
# Example:
# ssl-password = c5esWL7s

ssl-password = 24^gold

## Database connection information.
#
# Required if the database is an external (not embedded) database.

# db-user =
# db-password =
# db-host =
# db-sid =
# db-port = 1521
# db-protocol = TCP

## The location (absolute path) of the satellite certificate file.
#  Required.
#
# Example:
# satellite-cert-file = /tmp/satcert.cert

satellite-cert-file = /root/resources/redhat-internal-5.3.cert

## Apache conf.d/ssl.conf virtual host definition reconfiguration
#
# A value of "Y" or "y" here causes the installer to make a numbered
# backup of the system's existing httpd/conf.d/ssl.conf file and replace
# the original with one that's set up properly to work with Spacewalk.
# The recommended answer is Y
#
```

```
# ssl-config-sslvhost =
ssl-config-sslvhost = Y

# *** Options below this line usually don't need to be set. ***

# The Satellite server's hostname.  This must be the working FQDN of
# the satellite server.
#
hostname = sat-vm.cloud.lab.eng.bos.redhat.com

# The mount point for the RHN package repository.  Defaults to
# /var/rhn/satellite
#
# mount-point =

# Mail configuration.
#
# mail-mx =
# mdom =

# 'Common name' for the SSL certificates.  Defaults to the system's
# hostname, or whatever 'hostname' is set to.
#
# ssl-set-common-name =

# The email address for the SSL certificates.  Defaults to 'admin-email'.
#
ssl-set-email = <username>@<company>.com

# The expiration (in years) for the satellite certificates.  Defaults
# to the number of years until 2037.
#
# ssl-ca-cert-expiration =
# ssl-server-cert-expiration =

# Set to 'yes' to automatically install needed packages from RHN, provided the
# system is registered. Set to 'no' to terminate the installer if any needed
# packages are missing. Default is to prompt.
#
run-updater = yes

# *** For troubleshooting/testing only. ***
#
# rhn-parent =
# ssl-dir =
# ssl-server-rpm =
```

A.2.2 AppPGP

When generating the application channel security key for satellite, the input to gpg is specified in this file.

```
%echo Generating a standard key
Key-Type: DSA
Key-Length: 1024
Subkey-Type: ELG-E
Subkey-Length: 1024
Name-Real: Vijay Trehan
Name-Comment: Cloud Foundations
Name-Email: sm@redhat.com
Expire-Date: 0
Passphrase: Cloud Foundations
# Do a commit here, so that we can later print "done" :-)
%commit
%echo done
```

A.2.3 DNS database files

Nine files exist for the named/DNS database. The forward DNS file name is db.cloud.lab.eng.bos.redhat.com, the partial contents of which are listed below.

```
$TTL 86400
@       SOA     sat-vm milo.redhat.com. ( 9876 10800 3600 604800 600 )
        NS      sat-vm
     NS     mgmt1.cloud.lab.eng.bos.redhat.com.
     NS     mgmt2.cloud.lab.eng.bos.redhat.com.


localhost               A       127.0.0.1
sat-vm                  A       10.16.136.1
mgmt1                   A       10.16.136.10
mgmt2                   A       10.16.136.15
[ . . . ]
cloud-143-247           A       10.16.143.247
cloud-143-248           A       10.16.143.248
cloud-143-249           A       10.16.143.249
```

The reverse DNS file names are db.10.16.{136,137,138,139,140,141,142,143} and have contents similar to:

```
$TTL 86400
@       SOA     sat-vm.cloud.lab.eng.bos.redhat.com. ( 9876 10800 3600 604800 600 )
        NS      sat-vm.cloud.lab.eng.bos.redhat.com.
     NS     mgmt1.cloud.lab.eng.bos.redhat.com.
     NS     mgmt2.cloud.lab.eng.bos.redhat.com.


1               PTR     cloud-138-1.cloud.lab.eng.bos.redhat.com.
2               PTR     cloud-138-2.cloud.lab.eng.bos.redhat.com.
```

```
3                PTR     cloud-138-3.cloud.lab.eng.bos.redhat.com.
[ . . . ]
253              PTR     cloud-138-253.cloud.lab.eng.bos.redhat.com.
254              PTR     cloud-138-254.cloud.lab.eng.bos.redhat.com.
255              PTR     cloud-138-255.cloud.lab.eng.bos.redhat.com.
```

A.2.4 dhcpd.conf

This file is placed in */etc/dhcpd.conf*.

```
#
# DHCP Server Configuration file.
#   see /usr/share/doc/dhcp*/dhcpd.conf.sample
#
authoritive;
ddns-update-style interim;
ignore client-updates;

subnet 10.16.136.0 netmask 255.255.248.0 {
        option routers                  10.16.143.254;
        option subnet-mask              255.255.248.0;
        option domain-name              "cloud.lab.eng.bos.redhat.com";
        option domain-name-servers      10.16.136.10,10.16.136.15,10.16.136.1;
        option time-offset              -18000; # Eastern Standard Time
        option ntp-servers              10.16.136.10,10.16.136.15;
        filename "pxelinux.0";
        range dynamic-bootp 10.16.138.1 10.16.143.249;
        next-server 10.16.136.1;

        default-lease-time 21600;
        max-lease-time 43200;

host rhevm-vm {
            option host-name "rhevm-vm.cloud.lab.eng.bos.redhat.com";
            hardware ethernet 52:54:00:C0:DE:33;
            fixed-address 10.16.136.40;
           }
host ra-c7000-01-db1-ilo {
            option host-name "ra-c7000-01-db1-ilo.cloud.lab.eng.bos.redhat.com";
            hardware ethernet D8:D3:85:5F:28:56;
            fixed-address 10.16.136.231;
           }
host ra-c7000-01-db2-ilo {
            option host-name "ra-c7000-01-db2-ilo.cloud.lab.eng.bos.redhat.com";
            hardware ethernet D8:D3:85:63:FF:1E;
            fixed-address 10.16.136.232;
           }
host ra-c7000-01-db3-ilo {
            option host-name "ra-c7000-01-db3-ilo.cloud.lab.eng.bos.redhat.com";
            hardware ethernet D8:D3:85:63:FF:E2;
            fixed-address 10.16.136.233;
        }
host ra-c7000-01-db4-ilo {
        option host-name "ra-c7000-01-db4-ilo.cloud.lab.eng.bos.redhat.com";
        hardware ethernet D8:D3:85:63:FF:74;
        fixed-address 10.16.136.234;
        }
host ra-c7000-01-db5-ilo {
        option host-name "ra-c7000-01-db5-ilo.cloud.lab.eng.bos.redhat.com";
        hardware ethernet D8:D3:85:63:DE:92;
        fixed-address 10.16.136.235;
        }
host ra-c7000-01-db6-ilo {
        option host-name "ra-c7000-01-db6-ilo.cloud.lab.eng.bos.redhat.com";
        hardware ethernet D8:D3:85:63:EE:DA;
        fixed-address 10.16.136.236;
        }
host ra-c7000-01-db7-ilo {
        option host-name "ra-c7000-01-db7-ilo.cloud.lab.eng.bos.redhat.com";
        hardware ethernet D8:D3:85:63:FF:2A;
        fixed-address 10.16.136.237;
        }
host ra-c7000-01-db8-ilo {
        option host-name "ra-c7000-01-db8-ilo.cloud.lab.eng.bos.redhat.com";
        hardware ethernet D8:D3:85:63:36:CC;
        fixed-address 10.16.136.238;
        }
host ra-c7000-01-db9-ilo {
        option host-name "ra-c7000-01-db9-ilo.cloud.lab.eng.bos.redhat.com";
        hardware ethernet D8:D3:85:63:CE:82;
        fixed-address 10.16.136.239;
        }
host ra-c7000-01-db10-ilo {
        option host-name "ra-c7000-01-db10-ilo.cloud.lab.eng.bos.redhat.com";
        hardware ethernet D8:D3:85:63:FF:5A;
        fixed-address 10.16.136.240;
        }
host ra-c7000-01-db11-ilo {
        option host-name "ra-c7000-01-db11-ilo.cloud.lab.eng.bos.redhat.com";
        hardware ethernet D8:D3:85:63:9E:8E;
        fixed-address 10.16.136.241;
        }
host ra-c7000-01-db12-ilo {
        option host-name "ra-c7000-01-db12-ilo.cloud.lab.eng.bos.redhat.com";
        hardware ethernet D8:D3:85:63:BE:0A;
        fixed-address 10.16.136.242;
        }
host ra-c7000-01-db13-ilo {
        option host-name "ra-c7000-01-db13-ilo.cloud.lab.eng.bos.redhat.com";
        hardware ethernet D8:D3:85:63:DE:B2;
        fixed-address 10.16.136.243;
        }
host ra-c7000-01-db14-ilo {
        option host-name "ra-c7000-01-db14-ilo.cloud.lab.eng.bos.redhat.com";
        hardware ethernet D8:D3:85:63:EE:26;
        fixed-address 10.16.136.244;
        }
host ra-c7000-01-db15-ilo {
        option host-name "ra-c7000-01-db15-ilo.cloud.lab.eng.bos.redhat.com";
        hardware ethernet D8:D3:85:63:FF:1A;
        fixed-address 10.16.136.245;
        }
host ra-c7000-01-db16-ilo {
        option host-name "ra-c7000-01-db16-ilo.cloud.lab.eng.bos.redhat.com";
        hardware ethernet D8:D3:85:5F:49:3E;
        fixed-address 10.16.136.246;
        }
host ra-msa20 {
        option host-name "ra-msa20.cloud.lab.eng.bos.redhat.com";
        hardware ethernet 00:C0:FF:DA:B7:F4;
        fixed-address 10.16.136.248;
        }
host ra-pc5400-01 {
        option host-name "ra-pc5400-01.cloud.lab.eng.bos.redhat.com";
        hardware ethernet 00:26:f1:1b:06:00;
        fixed-address 10.16.136.248;
        }
}
```

A.2.5 named.conf

The */etc/named.conf* file is used by the satellite to specify it as the master.

```
options {
        directory "/var/named";
        forwarders { 10.16.255.2 port 53; 10.16.255.3 port 53; };
        notify yes;
};

zone "cloud.lab.eng.bos.redhat.com." { type master;     file "db.cloud.lab.eng.bos.redhat.com"; };
zone "136.16.10.in-addr.arpa."  { type master;  file "db.10.16.136"; };
zone "137.16.10.in-addr.arpa."  { type master;  file "db.10.16.137"; };
zone "138.16.10.in-addr.arpa."  { type master;  file "db.10.16.138"; };
zone "139.16.10.in-addr.arpa."  { type master;  file "db.10.16.139"; };
zone "140.16.10.in-addr.arpa."  { type master;  file "db.10.16.140"; };
zone "141.16.10.in-addr.arpa."  { type master;  file "db.10.16.141"; };
zone "142.16.10.in-addr.arpa."  { type master;  file "db.10.16.142"; };
zone "143.16.10.in-addr.arpa."  { type master;  file "db.10.16.143"; };
```

A.2.6 resolv.conf

An */etc/resolv.conf* was put in place on the satellite.

```
search cloud.lab.eng.bos.redhat.com bos.redhat.com
nameserver 10.16.136.10
nameserver 10.16.136.1
nameserver 10.16.255.2
```

A.2.7 settings

Cobbler's configuration is specified by */etc/cobbler/settings*.

```
---
# cobbler settings file
# restart cobblerd and run "cobbler sync" after making changes
# This config file is in YAML 1.0 format
# see http://yaml.org
# ========================================================
# if 1, cobbler allows insertions of system records that duplicate
# the hostname information of other system records.  In general,
# this is undesirable.
allow_duplicate_hostnames: 0

# if 1, cobbler allows insertions of system records that duplicate
# the ip address information of other system records.  In general,
# this is undesirable.
allow_duplicate_ips: 0

# if 1, cobbler allows insertions of system records that duplicate
# the mac address information of other system records.  In general,
# this is undesirable.
allow_duplicate_macs: 0

# the path to BIND's executable for this distribution.
bind_bin: /usr/sbin/named

# Email out a report when cobbler finishes installing a system.
# enabled: set to 1 to turn this feature on
# sender: optional
# email: which addresses to email
# smtp_server: used to specify another server for an MTA
# subject: use the default subject unless overridden

build_reporting_enabled: 1
build_reporting_sender: ""
build_reporting_email: [ 'root@localhost' ]
build_reporting_smtp_server: "localhost"
build_reporting_subject: ""

# Cheetah-language kickstart templates can import Python modules.
# while this is a useful feature, it is not safe to allow them to
# import anything they want. This whitelists which modules can be
# imported through Cheetah.  Users can expand this as needed but
# should never allow modules such as subprocess or those that
# allow access to the filesystem as Cheetah templates are evaluated
# by cobblerd as code.
cheetah_import_whitelist:
  - "random"
  - "re"
  - "time"

# if no kickstart is specified, use this template (FIXME)
default_kickstart: /var/lib/cobbler/kickstarts/default.ks

# cobbler has various sample kickstart templates stored
# in /var/lib/cobbler/kickstarts/.  This controls
# what install (root) password is set up for those
# systems that reference this variable.  The factory
# default is "cobbler" and cobbler check will warn if
# this is not changed.

default_password_crypted: "$1$1o751Xnc$kmQKHj6gtZ50IILNkHkkF0"

# configure all installed systems to use these nameservers by default
# unless defined differently in the profile.  For DHCP configurations
# you probably do /not/ want to supply this.
default_name_servers: []

# for libvirt based installs in koan, if no virt bridge
# is specified, which bridge do we try?  For EL 4/5 hosts
# this should be xenbr0, for all versions of Fedora, try
# "virbr0".  This can be overridden on a per-profile
# basis or at the koan command line though this saves
# typing to just set it here to the most common option.
default_virt_bridge: cloud0

# if koan is invoked without --virt-type and no virt-type
# is set on the profile/system, what virtualization type
# should be assumed?  Values: xenpv, xenfv, qemu, vmware
# (NOTE: this does not change what virt_type is chosen by import)
default_virt_type: qemu

# use this as the default disk size for virt guests (GB)
default_virt_file_size: 20

# use this as the default memory size for virt guests (MB)
default_virt_ram: 2048

# if using the authz_ownership module (see the Wiki), objects
# created without specifying an owner are assigned to this
# owner and/or group.  Can be a comma separated list.
default_ownership:
   - "admin"
```

```
# controls whether cobbler will add each new profile entry to the default
# PXE boot menu.  This can be over-ridden on a per-profile
# basis when adding/editing profiles with --enable-menu=0/1.  Users
# should ordinarily leave this setting enabled unless they are concerned
# with accidental reinstalls from users who select an entry at the PXE
# boot menu.  Adding a password to the boot menus templates
# may also be a good solution to prevent unwanted reinstallations
enable_menu: 1

# location for some important binaries and config files
# that can vary based on the distribution.
dhcpd_bin: /usr/sbin/dhcpd
dhcpd_conf: /etc/dhcpd.conf
dnsmasq_bin: /usr/sbin/dnsmasq
dnsmasq_conf: /etc/dnsmasq.conf

# enable Func-integration?  This makes sure each installed machine is set up
# to use func out of the box, which is a powerful way to script and control
# remote machines.
# Func lives at http://fedorahosted.org/func
# read more at https://fedorahosted.org/cobbler/wiki/FuncIntegration
# you will need to mirror Fedora/EPEL packages for this feature, so see
# https://fedorahosted.org/cobbler/wiki/ManageYumRepos if you want cobbler
# to help you with this

func_auto_setup: 0
func_master: overlord.example.org

# more important file locations...
httpd_bin: /usr/sbin/httpd

# change this port if Apache is not running plaintext on port
# 80.  Most people can leave this alone.
http_port: 80

# kernel options that should be present in every cobbler installation.
# kernel options can also be applied at the distro/profile/system
# level.
kernel_options:
    ksdevice: bootif
    lang: ' '
    text: ~

# s390 systems require additional kernel options in addition to the
# above defaults

kernel_options_s390x:
    RUNKS: 1
    ramdisk_size: 40000
    root: /dev/ram0
    ro: ~
    ip: off
    vnc: ~

# configuration options if using the authn_ldap module. See the
# the Wiki for details.  This can be ignored if you are not using
# LDAP for WebUI/XMLRPC authentication.
ldap_server: "ldap.example.com"
ldap_base_dn: "DC=example,DC=com"
ldap_port: 389
ldap_tls: 1
ldap_anonymous_bind: 1
ldap_search_bind_dn: "
ldap_search_passwd: "
ldap_search_prefix: 'uid='

# set to 1 to enable Cobbler's DHCP management features.
# the choice of DHCP management engine is in /etc/cobbler/modules.conf
manage_dhcp: 1

# set to 1 to enable Cobbler's DNS management features.
# the choice of DNS management engine is in /etc/cobbler/modules.conf
manage_dns: 1

# if using BIND (named) for DNS management in /etc/cobbler/modules.conf
# and manage_dns is enabled (above), this lists which zones are managed
# See the Wiki (https://fedorahosted.org/cobbler/wiki/ManageDns) for more info
manage_forward_zones:
    - 'cloud.lab.eng.bos.redhat.com'
manage_reverse_zones:
    - '10.16.136'
    - '10.16.137'
    - '10.16.138'
    - '10.16.139'
    - '10.16.140'
    - '10.16.141'
    - '10.16.142'
    - '10.16.143'

# cobbler has a feature that allows for integration with config management
# systems such as Puppet.  The following parameters work in conjunction with
# --mgmt-classes  and are described in further detail at:
# https://fedorahosted.org/cobbler/wiki/UsingCobblerWithConfigManagementSystem
mgmt_classes: []
mgmt_parameters:
   from_cobbler: 1

# location where cobbler will write its named.conf when BIND dns management is
# enabled
named_conf: /etc/named.conf
```

```
# if using cobbler with manage_dhcp, put the IP address
# of the cobbler server here so that PXE booting guests can find it
# if you do not set this correctly, this will be manifested in TFTP open timeouts.
next_server: sat-vm.cloud.lab.eng.bos.redhat.com

# if using cobbler with manage_dhcp and ISC, omapi allows realtime DHCP
# updates without restarting ISC dhcpd.  However, it may cause
# problems with removing leases and make things less reliable. OMAPI
# usage is experimental and not recommended at this time.

omapi_enabled: 0
omapi_port: 647
omshell_bin: /usr/bin/omshell

# settings for power management features.  optional.
# see https://fedorahosted.org/cobbler/wiki/PowerManagement to learn more
# choices:
#    bullpap
#    wti
#    apc_snmp
#    ether-wake
#    ipmilan
#    drac
#    ipmitool
#    ilo
#    rsa
#    lpar
#    bladecenter
#    virsh

power_management_default_type: 'ilo'

# the commands used by the power management module are sourced
# from what directory?
power_template_dir: "/etc/cobbler/power"

# if this setting is set to 1, cobbler systems that pxe boot
# will request at the end of their installation to toggle the
# --netboot-enabled record in the cobbler system record.  This eliminates
# the potential for a PXE boot loop if the system is set to PXE
# first in it's BIOS order.  Enable this if PXE is first in your BIOS
# boot order, otherwise leave this disabled.   See the manpage
# for --netboot-enabled.
pxe_just_once: 0

# the templates used for PXE config generation are sourced
# from what directory?
pxe_template_dir: "/etc/cobbler/pxe"
```

```
# Are you using a Red Hat management platform in addition to Cobbler?
# Cobbler can help you register to it.  Choose one of the following:
#   "off"    : I'm not using Red Hat Network, Satellite, or Spacewalk
#   "hosted" : I'm using Red Hat Network
#   "site"   : I'm using Red Hat Satellite Server or Spacewalk
# You will also want to read: https://fedorahosted.org/cobbler/wiki/TipsForRhn

redhat_management_type: "site"

# if redhat_management_type is enabled, choose your server
#   "management.example.org" : For Satellite or Spacewalk
#   "xmlrpc.rhn.redhat.com"  : For Red Hat Network
# This setting is also used by the code that supports using Spacewalk/Satellite users/passwords
# within Cobbler Web and Cobbler XMLRPC.  Using RHN Hosted for this is not supported.
# This feature can be used even if redhat_management_type is off, you just have
# to have authn_spacewalk selected in modules.conf

redhat_management_server: "sat-vm.cloud.lab.eng.bos.redhat.com"

# specify the default Red Hat authorization key to use to register
# system.  If left blank, no registration will be attempted.  Similarly
# you can set the --redhat-management-key to blank on any system to
# keep it from trying to register.
redhat_management_key: ""

# if using authn_spacewalk in modules.conf to let cobbler authenticate
# against Satellite/Spacewalk's auth system, by default it will not allow per user
# access into Cobbler Web and Cobbler XMLRPC.
# in order to permit this, the following setting must be enabled HOWEVER
# doing so will permit all Spacewalk/Satellite users of certain types to edit all
# of cobbler's configuration.
# these roles are:  config_admin and org_admin
# users should turn this on only if they want this behavior and
# do not have a cross-multi-org separation concern.  If you have
# a single org in your satellite, it's probably safe to turn this
# on and then you can use CobblerWeb alongside a Satellite install.
redhat_management_permissive: 1

# when DHCP and DNS management are enabled, cobbler sync can automatically
# restart those services to apply changes.  The exception for this is
# if using ISC for DHCP, then omapi eliminates the need for a restart.
# omapi, however, is experimental and not recommended for most configurations.
# If DHCP and DNS are going to be managed, but hosted on a box that
# is not on this server, disable restarts here and write some other
# script to ensure that the config files get copied/rsynced to the destination
# box.  This can be done by modifying the restart services trigger.
# Note that if manage_dhcp and manage_dns are disabled, the respective
# parameter will have no effect.  Most users should not need to change
# this.
restart_dns: 1
```

```
restart_dhcp: 1

# if set to 1, allows /usr/bin/cobbler-register (part of the koan package)
# to be used to remotely add new cobbler system records to cobbler.
# this effectively allows for registration of new hardware from system
# records.
register_new_installs: 1

# install triggers are scripts in /var/lib/cobbler/triggers/install
# that are triggered in kickstart pre and post sections.  Any
# executable script in those directories is run.  They can be used
# to send email or perform other actions.  They are currently
# run as root so if you do not need this functionality you can
# disable it, though this will also disable "cobbler status" which
# uses a logging trigger to audit install progress.
run_install_triggers: 1

# enables a trigger which version controls all changes to /var/lib/cobbler
# when add, edit, or sync events are performed.  This can be used
# to revert to previous database versions, generate RSS feeds, or for
# other auditing or backup purposes.  git is the recommend SCM
# for use with this feature.

scm_track_enabled: 0
scm_track_mode: "git"

# this is the address of the cobbler server -- as it is used
# by systems during the install process, it must be the address
# or hostname of the system as those systems can see the server.
# if you have a server that appears differently to different subnets
# (dual homed, etc), you need to read the --server-override section
# of the manpage for how that works.
server: sat-vm.cloud.lab.eng.bos.redhat.com

# this is a directory of files that cobbler uses to make
# templating easier.  See the Wiki for more information.  Changing
# this directory should not be required.
snippetsdir: /var/lib/cobbler/snippets

# by default, installs are *not* set to send installation logs to the cobbler
# server.  With 'anamon_enabled', kickstart templates may use the pre_anamon
# snippet to allow remote live monitoring of their installations from the
# cobbler server.  Installation logs will be stored under
# /var/log/cobbler/anamon/.  NOTE: This does allow an xmlrpc call to send logs
# to this directory, without authentication, so enable only if you are
# ok with this limitation.
anamon_enabled: 1

# locations of the TFTP binary and config file
tftpd_bin: /usr/sbin/in.tftpd
```

```
tftpd_conf: /etc/xinetd.d/tftp

# cobbler's web directory.  Don't change this setting -- see the
# Wiki on "relocating your cobbler install" if your /var partition
# is not large enough.
webdir: /var/www/cobbler

# cobbler's public XMLRPC listens on this port.  Change this only
# if absolutely needed, as you'll have to start supplying a new
# port option to koan if it is not the default.
xmlrpc_port: 25151

# "cobbler repo add" commands set cobbler up with repository
# information that can be used during kickstart and is automatically
# set up in the cobbler kickstart templates.  By default, these
# are only available at install time.  To make these repositories
# usable on installed systems (since cobbler makes a very convenient)
# mirror, set this to 1.  Most users can safely set this to 1.  Users
# who have a dual homed cobbler server, or are installing laptops that
# will not always have access to the cobbler server may wish to leave
# this as 0.  In that case, the cobbler mirrored yum repos are still
# accessible at http://cobbler.example.org/cblr/repo_mirror and yum
# configuration can still be done manually.  This is just a shortcut.
yum_post_install_mirror: 1

# additional flags to yum commands
yumreposync_flags: "-l"
yumdownloader_flags: "--resolve"
```

A.3 RHEV Image File

Creating the RHEV image is comprised of the steps below, which also guide the installation of the VM during the build process.

1. Create LVM volume for install

   ```
   lvcreate -n RHEVMvol -L 30G MgmtServicesVG
   ```

2. Create VM

   ```
   virt-install -n RHEVM -r 4096 --vcpus=2 --cpuset=auto --os-type=windows
   --os-variant=win2k8 --accelerate --network=bridge:cloud0
   --mac=52:54:00:C0:DE:01 --vnc --disk path=/dev/MgmtServicesVG/RHEVMvol
   --disk
   path=/pub/RHEV/kits/en_windows_server_2008_r2_standard_enterprise_datacenter_web_retail_x64_dvd_x15-50365.iso,device=cdrom,bus=ide
   ```

   or

   ```
   virt-manager
   ```

3. Start the VM console

4. Start Windows installation

   a) Select localization preferences

b) Select *Install now*

c) Choose OS Version (e.g., *Windows Server 2008 R2 Enterprise (Full Installation)*)

d) Accept License terms

e) Choose Custom install

f) Choose Disk size (e.g., 30GB)

g) Upon reboot after Windows installation, set Administrator password

h) Optionally, activate Windows

i) Set the time zone

j) Configure networking

- Switch CD/DVD to RHEV-tools ISO image
- Install virtio drivers (e.g., *RHEV-Network64*)
- Set TCP/IP properties

k) Configure/Install updates

l) Install .NET Framework features

m) Optionally enable Remote Desktop

n) Enable Windows Firewall

- Enable existing rule - “File and Printer Sharing (echo Request – ICMPv4-in)”
- Confirm rule enabled - “ WWWService (http traffic-in)”
- Confirm rule enables - “WWWService (https traffic-in)”
- Add rule – inbound TCP Ports 8006-8009

o) Optionally, change screen resolution

p) Create user 'admin' and add to the Administrators group

q) Optionally disable Enhanced Security Configuration

r) Create and populate the *C:\saved* folder

- Download RHEV-M software from Red Hat Network
- Copy RHEV-M installation answer file
- Copy *add-vms.ps1* script
- Copy *RHEVMInitConfig.ps1* script
- Optionally, copy/download user preferred applications (e.g., putty, notepad++)

s) Perform a second Windows Update

t) Shutdown to change the network definition of the VM

5. On the host, change the VM's network to *virtio*

a) Edit definition file (e.g., */etc/libvirt/qemu/RHEVM.xml*) such that the network stanza includes the virtio model type.

```
<interface type='bridge'>
```

```
    <mac address='52:54:00:c0:de:01'/>
    <source bridge='cloud0'/>
    <model type='virtio'/>
  </interface>
```

b) Update libvirtd with changes

```
virsh define /etc/libvirt/qemu/RHEVM.xml
```

c) Start vm

```
virsh start RHEVM
```

6. Set Network Properties
7. Perform System Preparation (*\Windows\System32\sysprep\sysprep.exe*)
   - System Cleanup Actions: *Enter System Out-of-Box-Experience*
   - Select the *Generalize* checkbox
   - Shutdown Options: *Shutdown*
8. On host, create image file

```
dd if=/dev/MgmtServicesVG/RHEVMvol of=/rhevm-`date +%y%m%d`.img bs=1024k
```

9. On the host, compress the image file

```
bzip2 /rhevm-`date +%y%m%d`.img
```

A.4 MRG

A.4.1 condor_config.local.mgr

This is the MRG Grid configuration file for the MRG Manager VM and is downloaded to */var/lib/condor/condor_config.local*.

```
# This config disables advertising to UW's world collector. Changing
# this config option will have your pool show up in UW's world
# collector and eventually on the world map of Condor pools.
CONDOR_DEVELOPERS = NONE

# What the relay looks like from the inside.
PRIVATE_HOST = <internal IP>
# What the relay looks like from the outside.
PUBLIC_HOST = <elastic IP>
PUBLIC_PORT = 9618

# Accept TCP updates, necessary because default is UDP
# and the relay tunnel is only TCP
COLLECTOR_SOCKET_CACHE_SIZE = 1024

# Setting CCB_ADDRESS results in deadlock because
# the Collector does not realize it is making a blocking
# connection to itself via the CCB_ADDRESS. So, enable CCB
# for only the Shadow, which needs to receive a connection
# from the Starter for file transfer.
```

```
CCB_ADDRESS = $(PUBLIC_HOST):$(PUBLIC_PORT)
COLLECTOR.CCB_ADDRESS =

# Avoid needing CCB within the VPN
PRIVATE_NETWORK_NAME = mrg-vm

# Set TCP_FORWARDING_HOST so CCB will advertise its public
# address. Without this, it will advertise its private address
# and the Starter will not be able to connect to reverse its
# connection to the Shadow.
COLLECTOR.TCP_FORWARDING_HOST = $(PUBLIC_HOST)
# As per TCP_FORWARDING_HOST semantics, the local port for
# the Collector/CCB must match the relay port
COLLECTOR_HOST = $(FULL_HOSTNAME):$(PUBLIC_PORT)

# Give access to relayed communication
ALLOW_WRITE = $(ALLOW_WRITE), $(PRIVATE_HOST)

CONDOR_HOST = $(FULL_HOSTNAME)
COLLECTOR_NAME = Grid On a Cloud
#COLLECTOR_HOST = $(CONDOR_HOST)
NEGOTIATOR_HOST = $(CONDOR_HOST)
UID_DOMAIN = cloud.lab.eng.bos.redhat.com
FILESYSTEM_DOMAIN = cloud.lab.eng.bos.redhat.com
START = TRUE
SUSPEND = FALSE
PREEMPT = FALSE
KILL = FALSE
HOSTALLOW_WRITE = *.cloud.lab.eng.bos.redhat.com,  $(PRIVATE_HOST)
HOSTALLOW_READ = *.cloud.lab.eng.bos.redhat.com
DAEMON_LIST = COLLECTOR, MASTER, NEGOTIATOR, SCHEDD
NEGOTIATOR_INTERVAL = 20
TRUST_UID_DOMAIN = TRUE
IN_HIGHPORT = 9800
IN_LOWPORT  = 9600

SCHEDD.PLUGINS = $(LIB)/plugins/MgmtScheddPlugin-plugin.so
COLLECTOR.PLUGINS = $(LIB)/plugins/MgmtCollectorPlugin-plugin.so
NEGOTIATOR.PLUGINS = $(LIB)/plugins/MgmtNegotiatorPlugin-plugin.so

# Plugin configuration
MASTER.PLUGINS = $(LIB)/plugins/MgmtMasterPlugin-plugin.so
QMF_BROKER_HOST = mrg-vm.cloud.lab.eng.bos.redhat.com
```

A.4.2 condor_config.local_mrgexec

This version of the file is use by generic MRG execute nodes.

```
# This config disables advertising to UW's world collector. Changing
# this config option will have your pool show up in UW's world
# collector and eventually on the world map of Condor pools.
```

```
CONDOR_DEVELOPERS = NONE

CONDOR_HOST = mrg-vm.cloud.lab.eng.bos.redhat.com
COLLECTOR_HOST = $(CONDOR_HOST)
COLLECTOR_NAME = Grid On a Cloud
FILESYSTEM_DOMAIN = $(FULL_HOSTNAME)
UID_DOMAIN = $(FULL_HOSTNAME)

START = TRUE
SUSPEND = FALSE
PREEMPT = FALSE
KILL = FALSE
ALLOW_WRITE = $(FULL_HOSTNAME), *.cloud.lab.eng.bos.redhat.com
DAEMON_LIST = MASTER, STARTD
NEGOTIATOR_INTERVAL = 20
TRUST_UID_DOMAIN = TRUE
IN_HIGHPORT = 9700
IN_LOWPORT  = 9600

# Plugin configuration
MASTER.PLUGINS = $(LIB)/plugins/MgmtMasterPlugin-plugin.so
QMF_BROKER_HOST = mrg-vm.cloud.lab.eng.bos.redhat.com
```

A.4.3 condor_config.local_mrgexecrender

This version of the MRG configuration file is for MRG Execute nodes configured to rendering movies and/or pictures.

```
# This config disables advertising to UW's world collector. Changing
# this config option will have your pool show up in UW's world
# collector and eventually on the world map of Condor pools.
CONDOR_DEVELOPERS = NONE

CONDOR_HOST = mrg-vm.cloud.lab.eng.bos.redhat.com
COLLECTOR_HOST = $(CONDOR_HOST)
COLLECTOR_NAME = Grid On a Cloud
FILESYSTEM_DOMAIN = cloud.lab.eng.bos.redhat.com
UID_DOMAIN = cloud.lab.eng.bos.redhat.com

START = TRUE
SUSPEND = FALSE
PREEMPT = FALSE
KILL = FALSE
ALLOW_WRITE = $(FULL_HOSTNAME), *.cloud.lab.eng.bos.redhat.com
DAEMON_LIST = MASTER, STARTD
NEGOTIATOR_INTERVAL = 20
TRUST_UID_DOMAIN = TRUE
IN_HIGHPORT = 9700
IN_LOWPORT  = 9600
```

```
# Plugin configuration
MASTER.PLUGINS = $(LIB)/plugins/MgmtMasterPlugin-plugin.so
QMF_BROKER_HOST = mrg-vm.cloud.lab.eng.bos.redhat.com
```

A.4.4 cumin.conf

This file is downloaded to */etc/cumin/cumin.conf*.

```
[main]
data: postgresql://cumin@localhost/cumin
addr: 10.16.136.50
ssl: yes
```

A.4.5 pg_hba.conf

```
# PostgreSQL Client Authentication Configuration File
# ===================================================
#
# Refer to the PostgreSQL Administrator's Guide, chapter "Client
# Authentication" for a complete description.  A short synopsis
# follows.
#
# This file controls: which hosts are allowed to connect, how clients
# are authenticated, which PostgreSQL user names they can use, which
# databases they can access.  Records take one of these forms:
#
# local      DATABASE  USER  METHOD  [OPTION]
# host       DATABASE  USER  CIDR-ADDRESS  METHOD  [OPTION]
# hostssl    DATABASE  USER  CIDR-ADDRESS  METHOD  [OPTION]
# hostnossl  DATABASE  USER  CIDR-ADDRESS  METHOD  [OPTION]
#
# (The uppercase items must be replaced by actual values.)
#
# The first field is the connection type: "local" is a Unix-domain socket,
# "host" is either a plain or SSL-encrypted TCP/IP socket, "hostssl" is an
# SSL-encrypted TCP/IP socket, and "hostnossl" is a plain TCP/IP socket.
#
# DATABASE can be "all", "sameuser", "samerole", a database name, or
# a comma-separated list thereof.
#
# USER can be "all", a user name, a group name prefixed with "+", or
# a comma-separated list thereof.  In both the DATABASE and USER fields
# you can also write a file name prefixed with "@" to include names from
# a separate file.
#
# CIDR-ADDRESS specifies the set of hosts the record matches.
# It is made up of an IP address and a CIDR mask that is an integer
# (between 0 and 32 (IPv4) or 128 (IPv6) inclusive) that specifies
# the number of significant bits in the mask.  Alternatively, you can write
# an IP address and netmask in separate columns to specify the set of hosts.
#
```

```
# METHOD can be "trust", "reject", "md5", "crypt", "password",
# "krb5", "ident", or "pam".  Note that "password" sends passwords
# in clear text; "md5" is preferred since it sends encrypted passwords.
#
# OPTION is the ident map or the name of the PAM service, depending on METHOD.
#
# Database and user names containing spaces, commas, quotes and other special
# characters must be quoted. Quoting one of the keywords "all", "sameuser" or
# "samerole" makes the name lose its special character, and just match a
# database or username with that name.
#
# This file is read on server startup and when the postmaster receives
# a SIGHUP signal.  If you edit the file on a running system, you have
# to SIGHUP the postmaster for the changes to take effect.  You can use  will
# "pg_ctl reload" to do that.

# Put your actual configuration here
# ----------------------------------
#
# If you want to allow non-local connections, you need to add more
# "host" records. In that case you will also need to make PostgreSQL listen
# on a non-local interface via the listen_addresses configuration parameter,
# or via the -i or -h command line switches.
#

# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD
host    cumin       cumin       127.0.0.1/32          trust
# "local" is for Unix domain socket connections only
local   all         all                               ident sameuser
# IPv4 local connections:
host    all         all         127.0.0.1/32          ident sameuser
# IPv6 local connections:
host    all         all         ::1/128               ident sameuser
```

A.4.6 Perfect

The perfect tar file was made by creating a compressed tar file of the perfect subdirectory. Details on the files can be found in the **Red Hat Cloud Foundations Edition One: Hybrid IaaS Cloud** document.

```
tar tf perfect.tgz
perfect/
perfect/perfect.py
perfect/mk_jobs.sh
perfect/output/
perfect/submit_jobs.sh
perfect/ec2_tunnel.sub
perfect/ec2_mrgexec.sub
perfect/perfect
```

A.4.7 Blender

Blender is an open source 3D content creation suite and is used in movie rendering example in this paper. Version 2.48a was used for compatibility with the version of python in Red Hat Enterprise Linux 5.5 and was download from *http://download.blender.org/release/Blender2.48a/blender-2.48a-linux-glibc236-py24-x86_64-static.tar.bz2*.

A.4.8 render.tgz

This render tar file contains the command scripts and source to render of a small movie clip. The movie clip is from “elephant's dream” and the source can be downloaded from *http://orange.blender.org/download*.

Other files included in the tar file but not from “elephant's dream” source are:

render/
render/setup_jobs.rb
render/production/
render/production/jobs/
render/production/jobs/start_job.sh
render/production/jobs/hello.sh
render/production/jobs/testjob

setup_jobs.rb - a ruby script which creates the submit files

```
#!/usr/bin/ruby

RENDER_HOME="/home/admin/render"
BLEND_FILE=RENDER_HOME+"/production/07_emo_flips_out/07_01.blend"
FRAME_DIR=RENDER_HOME+"/production/frames"
MOVIE_DIR=RENDER_HOME+"/production/movies"
LOGS_HOME=RENDER_HOME+"/production/logs"
JOBS_HOME=RENDER_HOME+"/production/jobs"
NUM_FRAMES=45

setupFile = File.new("%s/render_setup.sub" % [JOBS_HOME], "w")
setupFile.write("Universe = vanilla\n")
setupFile.write("Executable = %s/start_job.sh\n" % [RENDER_HOME])
setupFile.write("Log = %s/setup.log\n" % [LOGS_HOME])
setupFile.write("Output = %s/setup.out\n" % [LOGS_HOME])
setupFile.write("Error = %s/setup.err\n" % [LOGS_HOME])
setupFile.write("transfer_executable = false\n")
setupFile.write("Requirements = Arch =?= \"X86_64\"\n")
setupFile.write("QUEUE\n")

for frame in 1..NUM_FRAMES
   jobFile = File.new("%s/frame%d.sub" % [JOBS_HOME, frame], "w")
   jobFile.write("Universe = vanilla\n")
   jobFile.write("Executable = /home/admin/blender-2.48a-linux-glibc236-py24-x86_64-static/blender\n")
   jobFile.write("Arguments = -b %s -F PNG -o %s/frame_##.png -f %d\n" % \
         [BLEND_FILE, FRAME_DIR, frame])
   jobFile.write("Log = %s/frame%d.log\n" % [LOGS_HOME, frame])
   jobFile.write("Output = %s/frame%d.out\n" % [LOGS_HOME, frame])
   jobFile.write("Error = %s/frame%d.err\n" % [LOGS_HOME, frame])
   jobFile.write("transfer_executable = false\n")
   jobFile.write("Requirements = Arch =?= \"X86_64\"\n")
   jobFile.write("QUEUE\n")
   jobFile.close
end

movieFile = File.new("%s/create_movie.sub"%[JOBS_HOME], "w")
   movieFile.write("Universe = vanilla\n")
   movieFile.write("Executable = /usr/bin/mencoder\n")
   movieFile.write("Arguments = mf://%s/*.png -mf fps=5:type=png -ovc lavc -oac copy -o %s/output.avi\n" % [FRAME_DIR, MOVIE_DIR])
   movieFile.write("Log = %s/create_movie.log\n" % [LOGS_HOME])
   movieFile.write("Output = %s/create_movie.out\n" % [LOGS_HOME])
   movieFile.write("Error = %s/create_movie.err\n" % [LOGS_HOME])
   movieFile.write("Requirements = Arch =?= \"X86_64\"\n")
   movieFile.write("transfer_executable = false\n")
   movieFile.write("QUEUE\n")
movieFile.close

dagFile = File.new("%s/render.dag" % [JOBS_HOME], "w")
   dagFile.write("JOB Setup %s/testjob\n" % [JOBS_HOME] )
   for frame in 1..NUM_FRAMES
      dagFile.write("JOB Frame_%d frame%d.sub\n" % [frame, frame])
   end
   dagFile.write("JOB Create_Movie create_movie.sub\n")
   dagFile.write("PARENT Setup CHILD")
   for frame in 1..NUM_FRAMES
      dagFile.write(" Frame_%d" % [frame])
   end
   dagFile.write("\nPARENT")
   for frame in 1..NUM_FRAMES
      dagFile.write(" Frame_%d" % [frame])
   end
   dagFile.write(" CHILD Create_Movie\n")
dagFile.close
```

start_job.sh – script which clean up and current/previous jobs, then submits the job as prepared by *setup_jobs.rb*

```
#!/bin/bash
echo Removing all jobs in the queue ...
condor_rm -all
sleep 5
echo Cleaning up log/out/err files ....
rm -f /home/admin/render/production/logs/*.{log,out,err}
rm -f /home/admin/render/production/frames/frame*.*
rm -f /home/admin/render/production/jobs/render.dag.*
```

```
sleep 5
echo Sunmitting DAG ...
condor_submit_dag /home/admin/render/production/jobs/render.dag
```

hello.sh – simple script which write date of the start of the render process

```
#!/bin/bash

echo "`date '+%H:%M:%S'` Starting Render Job" >> /home/admin/render/production/logs/testloop.out
```

testjob – condor job description file used to submit the hello.sh script

```
#Test Job
Executable = /home/admin/render/production/jobs/hello.sh
Universe = vanilla
#input = test.data
#transfer_executable = true
#should_transfer_files = YES
#when_to_transfer_output = ON_EXIT
output = /home/admin/render/production/logs/testloop.out
error = /home/admin/render/production/logs/testloop.error
Log = /home/admin/render/production/logs/testloop.log
#args = if=/dev/zero of=/dev/null

queue
```

A.5 JBoss

All files related to JBoss and JON are described below.

A.5.1 JBoss EAP

jboss-eap-5.0.0.GA.zip is the zipfile containing the JBoss EAP installer which unzips to become the */root/resources/jboss-eap* directory. To make scripting easier, the softlink *jboss-eap-default.GA.zip* is maintained to reference the latest obtained zipfile for JBoss EAP, in this case v5.

A.5.2 JBoss Seam

The *jboss-seam-booking.ear* and *jboss-seam-booking-ds.xml* files comprise the JBoss Seam hotel booking demo which is deployed by the presence of both files in the */root/jboss-eap*/jboss-as/server/default/deploy* directory.

A.5.3 JON

JON in this environment is comprised of the files:

- *jon-server-2.4.0.GA.zip* – JON server installer
- *jon-license.xml* – JON license
- *jon-plugin-pack-eap-2.4.0.GA.zip* – plugin to allow monitoring of EAP instances
- *jon-plugin-pack-ews-2.4.0.GA.zip* – plugin for the EWS distribution of Tomcat

- *jon-plugin-pack-soa-2.4.0.GA.zip* – plugin supporting the SOA platform server
- *rhq-enterprise-agent-3.0.0.GA.jar* – contains the JON RHQ agent comprised of:
  - *enterprise-agent-default.GA.jar* – a softlink maintained to reference the latest obtained jarfile for the agent, in this case v3
  - *rhq-install.sh* – the agent installer
  - *rhq-agent-env.sh* - the init file residing on each JBoss instance reporting JBoss specific information to the JON server. Obtained by the kickstart, this environment configuration file is customized to define the following variables according to the working environment:
    - RHQ_AGENT_HOME
    - RHQ_AGENT_JAVA_HOME
    - RHQ_AGENT_CMDLINE_OPTS
    - RHQ_AGENT_PIDFILE_DIR

```
#=========================================================================
# RHQ Agent UNIX Startup Script Configuration File
#=========================================================================
#
#   RHQ_AGENT_DEBUG - If this is defined, the script will emit debug messages. It will also
#   enable debug messages to be emitted from the agent itself.
#   If not set or set to "false", debug is turned off. This does not implicitly enable Sigar native
#   debug mode. You must explicitly enable RHQ_AGENT_SIGAR_DEBUG in addition to
#   enabling RHQ_AGENT_DEBUG for Sigar logging to be enabled.
#
#RHQ_AGENT_DEBUG=true

#   RHQ_AGENT_SIGAR_DEBUG - Enables Sigar debug mode but only if agent debug
#                           is also enabled. See RHQ_AGENT_DEBUG for more.
#RHQ_AGENT_SIGAR_DEBUG=false

#   RHQ_AGENT_HOME - Defines where the agent's home install directory is.
#                    If not defined, it will be assumed to be the parent
#                    directory of the directory where this script lives.
#
RHQ_AGENT_HOME="/root/rhq-agent"

#   RHQ_AGENT_JAVA_HOME - The location of the JRE that the agent will
#                         use. This will be ignored if
#                         RHQ_AGENT_JAVA_EXE_FILE_PATH is set.
#                         If this and RHQ_AGENT_JAVA_EXE_FILE_PATH are
#                         not set, the agent's embedded JRE will be used.
#
RHQ_AGENT_JAVA_HOME="/usr/lib/jvm/jre-1.6.0"

#   RHQ_AGENT_JAVA_EXE_FILE_PATH - Defines the full path to the Java
#   executable to use. If this is set, RHQ_AGENT_JAVA_HOME is ignored.
#   If this is not set, then $RHQ_AGENT_JAVA_HOME/bin/java
#    is used. If this and RHQ_AGENT_JAVA_HOME are not set, the
#    agent's embedded JRE will be used.
#
#RHQ_AGENT_JAVA_EXE_FILE_PATH="/usr/local/bin/java"

#    RHQ_AGENT_JAVA_OPTS - Java VM command line options to be
#    passed into the agent's VM. If this is not defined this script will pass in a default set of options.
#    If this is set, it completely overrides the agent's defaults. If you only want to add options to the
#    agent's defaults, then you will want to use RHQ_AGENT_ADDITIONAL_JAVA_OPTS
#    instead.
#
#RHQ_AGENT_JAVA_OPTS="-Xms64m -Xmx128m -Djava.net.preferIPv4Stack=true"

#    RHQ_AGENT_JAVA_ENDORSED_DIRS - Java VM command line option to set the
#    endorsed dirs for the agent's VM. If this is not defined this script will pass in a
#    default value. If this is set, it completely overrides the agent's default.
#    However, if this is set to "none", the agent will not be passed the VM argument
#    to set the endorsed dirs.
#
#RHQ_AGENT_JAVA_ENDORSED_DIRS="${RHQ_AGENT_HOME}/lib/endorsed"

#    RHQ_AGENT_JAVA_LIBRARY_PATH - The RHQ Agent has a JNI library that
#    it needs to find in order to do things like execute PIQL queries and access
#    low-level operating system data. This is the native system layer (SIGAR).
#    If you deploy a custom plugin that also requires JNI libraries, you must add to
#    the library path here, but you must ensure not to remove the RHQ Agent library path.
#    If this variable is set, it completely overrides the agent's default.
#    However, if this is set to "none", the agent will not be passed the VM argument
#    to set the library paths.
#
#RHQ_AGENT_JAVA_LIBRARY_PATH="${RHQ_AGENT_HOME}/lib"

#    RHQ_AGENT_ADDITIONAL_JAVA_OPTS - additional Java VM command line options
#    to be passed into the agent's VM. This is added to RHQ_AGENT_JAVA_OPTS; it
#    is mainly used to augment the agent's default set of options. This can be
#    left unset if it is not needed.
#
#RHQ_AGENT_ADDITIONAL_JAVA_OPTS="-
agentlib:jdwp=transport=dt_socket,address=9797,server=y,suspend=n"

#    RHQ_AGENT_CMDLINE_OPTS - If this is defined, these are the command line
#    arguments that will be passed to the RHQ Agent. Any arguments specified on the command
#    line will be ignored. If this is not defined, the command line arguments given to the script are
#     passed through to the RHQ Agent. If you want to have command line arguments
#     added to the arguments specified here, append '$*' to the end of this option. For example,
#     "--daemon $*". In this case, both the command line options and the ones specified here will
#     be passed to the agent.
#
#RHQ_AGENT_CMDLINE_OPTS="--daemon --nonative --cleanconfig"
RHQ_AGENT_CMDLINE_OPTS="--daemon --cleanconfig"
```

```
#    RHQ_AGENT_IN_BACKGROUND - If this is defined, the RHQ Agent JVM will
#    be launched in the background (thus causing this script to exit immediately).  If the value is
#    something other than "nofile", it will be assumed to be a full file path which this script will
#    create and will contain the agent VM's process  pid value. If this is not defined, the VM is
#    launched in foreground and this script blocks until the VM exits, at which time this
#    script will also exit. NOTE: this should only be used by the rhq-agent-wrapper.sh script.
#    If you want to launch the agent in the background, use that script to do so instead of
#    setting this variable.
#
#RHQ_AGENT_IN_BACKGROUND=rhq-agent.pid

#=================================================================
# THE FOLLOWING ARE USED SOLELY FOR THE rhq-agent-wrapper.sh SCRIPT

#    RHQ_AGENT_PIDFILE_DIR - When rhq-agent-wrapper.sh is used to start
#    the agent, it runs the process in background and writes its pid to a pidfile. The default
#    location of this pidfile is in the agent's /bin directory. If you want to have the pidfile
#    written to another location, set this environment variable. This value must be a full path to a
#     directory with write permissions.
#
RHQ_AGENT_PIDFILE_DIR="/var/run"

#    RHQ_AGENT_START_COMMAND - If defined, this is the command that will be
#    executed to start the agent. Use this to customize how the agent process
#    is started (e.g., with "su" or "sudo"). This completely overrides the command used
#    to start the agent - you must ensure you provide a valid command that starts the agent
#    script 'rhq-agent.sh'. Note that if this start command requires the
#    user to enter a password, you can show a prompt to the user if you set the variable
#    RHQ_AGENT_PASSWORD_PROMPT. Also note that if your agent install directory
#    has spaces in its name, you might have to do some special string manipulation to get the
#    agent script to start. See below for an  example of how to do this.
#RHQ_AGENT_START_COMMAND="su -m -l user -c '${RHQ_AGENT_HOME}/bin/rhq-
# agent.sh'"
#RHQ_AGENT_START_COMMAND="su -m -l user -c '$(echo ${RHQ_AGENT_HOME}|sed 's/
# /\\ /')/bin/rhq-agent.sh'"

#    RHQ_AGENT_PASSWORD_PROMPT - if "true", this indicates that the user
#    that is to run the agent must type the password on the console in order to run.
#    Therefore, "true" forces a prompt message to appear on the console. If this is
#    defined, but not equal to "true", the value itself will be used as the prompt string.
#    This is not defined by default. Note that this setting does nothing more than simply
#    printing a message to the console.
#
#RHQ_AGENT_PASSWORD_PROMPT=true
```

A.6 Blade and Virtual Connect Configuration

The configuration of the blades as downloaded from the chassis On-board Administrator:

```
#Script Generated by user
#Generated on: Tue May 11 15:37:56 2010

#Set Enclosure Time
SET TIMEZONE EST5EDT

#Set Enclosure Information
SET ENCLOSURE ASSET TAG ""
SET ENCLOSURE NAME "ra-c7000-01"
SET RACK NAME "L2E6"
SET POWER MODE REDUNDANT
SET POWER SAVINGS ON

#Power limit must be within the range of 2700-16400
SET POWER LIMIT OFF
#Enclosure Dynamic Power Cap must be within the range of 2570-7822
SET ENCLOSURE POWER_CAP OFF
SET ENCLOSURE POWER_CAP_BAYS_TO_EXCLUDE None

#Set PowerDelay Information
SET INTERCONNECT POWERDELAY 1 0
SET INTERCONNECT POWERDELAY 2 0
SET INTERCONNECT POWERDELAY 3 0
SET INTERCONNECT POWERDELAY 4 0
SET INTERCONNECT POWERDELAY 5 0
SET INTERCONNECT POWERDELAY 6 0
SET INTERCONNECT POWERDELAY 7 0
SET INTERCONNECT POWERDELAY 8 0
SET SERVER POWERDELAY 1 0
SET SERVER POWERDELAY 2 0
SET SERVER POWERDELAY 3 0
SET SERVER POWERDELAY 4 0
SET SERVER POWERDELAY 5 0
SET SERVER POWERDELAY 6 0
SET SERVER POWERDELAY 7 0
SET SERVER POWERDELAY 8 0
SET SERVER POWERDELAY 9 0
SET SERVER POWERDELAY 10 0
SET SERVER POWERDELAY 11 0
SET SERVER POWERDELAY 12 0
SET SERVER POWERDELAY 13 0
SET SERVER POWERDELAY 14 0
SET SERVER POWERDELAY 15 0
SET SERVER POWERDELAY 16 0
SET SERVER POWERDELAY 1A 0
SET SERVER POWERDELAY 2A 0
SET SERVER POWERDELAY 3A 0
SET SERVER POWERDELAY 4A 0
SET SERVER POWERDELAY 5A 0
SET SERVER POWERDELAY 6A 0
SET SERVER POWERDELAY 7A 0
SET SERVER POWERDELAY 8A 0
SET SERVER POWERDELAY 9A 0
SET SERVER POWERDELAY 10A 0
SET SERVER POWERDELAY 11A 0
SET SERVER POWERDELAY 12A 0
SET SERVER POWERDELAY 13A 0
SET SERVER POWERDELAY 14A 0
SET SERVER POWERDELAY 15A 0
SET SERVER POWERDELAY 16A 0
SET SERVER POWERDELAY 1B 0
SET SERVER POWERDELAY 2B 0
SET SERVER POWERDELAY 3B 0
SET SERVER POWERDELAY 4B 0
SET SERVER POWERDELAY 5B 0
SET SERVER POWERDELAY 6B 0
SET SERVER POWERDELAY 7B 0
SET SERVER POWERDELAY 8B 0
SET SERVER POWERDELAY 9B 0
SET SERVER POWERDELAY 10B 0
SET SERVER POWERDELAY 11B 0
SET SERVER POWERDELAY 12B 0
SET SERVER POWERDELAY 13B 0
SET SERVER POWERDELAY 14B 0
SET SERVER POWERDELAY 15B 0
SET SERVER POWERDELAY 16B 0

#Configure Protocols
ENABLE WEB
ENABLE SECURESH
ENABLE TELNET
ENABLE XMLREPLY
ENABLE ENCLOSURE_IP_MODE
SET LLF INTERVAL 60
ENABLE LLF
ENABLE GUI_LOGIN_DETAIL

#Configure Alertmail
SET ALERTMAIL SMTPSERVER 0.0.0.0
DISABLE ALERTMAIL

#Configure Trusted Hosts
#REMOVE TRUSTED HOST ALL
DISABLE TRUSTED HOST

#Configure NTP
SET NTP PRIMARY 10.16.255.2
SET NTP SECONDARY 10.16.255.3
SET NTP POLL 720
ENABLE NTP
```

```
#Set SNMP Information
SET SNMP CONTACT ""
SET SNMP LOCATION ""
SET SNMP COMMUNITY READ "public"
SET SNMP COMMUNITY WRITE ""
DISABLE SNMP

#Set Remote Syslog Information
SET REMOTE SYSLOG SERVER ""
SET REMOTE SYSLOG PORT 514
DISABLE SYSLOG REMOTE

#Set Enclosure Bay IP Addressing (EBIPA) Information
SET EBIPA SERVER 10.16.136.231 1
SET EBIPA SERVER 10.16.136.232 2
SET EBIPA SERVER 10.16.136.233 3
SET EBIPA SERVER 10.16.136.234 4
SET EBIPA SERVER 10.16.136.235 5
SET EBIPA SERVER 10.16.136.236 6
SET EBIPA SERVER 10.16.136.237 7
SET EBIPA SERVER 10.16.136.238 8
SET EBIPA SERVER 10.16.136.239 9
SET EBIPA SERVER 10.16.136.240 10
SET EBIPA SERVER 10.16.136.241 11
SET EBIPA SERVER 10.16.136.242 12
SET EBIPA SERVER 10.16.136.243 13
SET EBIPA SERVER 10.16.136.244 14
SET EBIPA SERVER 10.16.136.245 15
SET EBIPA SERVER 10.16.136.246 16
SET EBIPA SERVER NONE 1A
SET EBIPA SERVER NONE 2A
SET EBIPA SERVER NONE 3A
SET EBIPA SERVER NONE 4A
SET EBIPA SERVER NONE 5A
SET EBIPA SERVER NONE 6A
SET EBIPA SERVER NONE 7A
SET EBIPA SERVER NONE 8A
SET EBIPA SERVER NONE 9A
SET EBIPA SERVER NONE 10A
SET EBIPA SERVER NONE 11A
SET EBIPA SERVER NONE 12A
SET EBIPA SERVER NONE 13A
SET EBIPA SERVER NONE 14A
SET EBIPA SERVER NONE 15A
SET EBIPA SERVER NONE 16A
SET EBIPA SERVER NONE 1B
SET EBIPA SERVER NONE 2B
SET EBIPA SERVER NONE 3B
SET EBIPA SERVER NONE 4B
SET EBIPA SERVER NONE 5B
SET EBIPA SERVER NONE 6B
SET EBIPA SERVER NONE 7B
SET EBIPA SERVER NONE 8B
SET EBIPA SERVER NONE 9B
SET EBIPA SERVER NONE 10B
SET EBIPA SERVER NONE 11B
SET EBIPA SERVER NONE 12B
SET EBIPA SERVER NONE 13B
SET EBIPA SERVER NONE 14B
SET EBIPA SERVER NONE 15B
SET EBIPA SERVER NONE 16B
SET EBIPA SERVER NETMASK 255.255.248.0
SET EBIPA SERVER GATEWAY 10.16.143.254
SET EBIPA SERVER DOMAIN "cloud.lab.eng.bos.redhat.com"
ADD EBIPA SERVER DNS 10.16.136.1
ADD EBIPA SERVER DNS 10.16.255.2
ADD EBIPA SERVER DNS 10.16.255.3
ENABLE EBIPA SERVER 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
DISABLE EBIPA SERVER
1A,2A,3A,4A,5A,6A,7A,8A,9A,10A,11A,12A,13A,14A,15A,16A,1B,2B,3B,4B,5B,6B,7B,8B,9B,10B,11B,12B,13B,14B,15B,16B
SET EBIPA INTERCONNECT 10.16.136.251 1
SET EBIPA INTERCONNECT 10.16.136.250 2
SET EBIPA INTERCONNECT 10.16.136.253 3
SET EBIPA INTERCONNECT 10.16.136.252 4
SET EBIPA INTERCONNECT NONE 5
SET EBIPA INTERCONNECT NONE 6
SET EBIPA INTERCONNECT NONE 7
SET EBIPA INTERCONNECT NONE 8
SET EBIPA INTERCONNECT NETMASK 255.255.248.0
SET EBIPA INTERCONNECT GATEWAY 10.16.143.254
SET EBIPA INTERCONNECT DOMAIN "cloud.lab.eng.bos.redhat.com"
ADD EBIPA INTERCONNECT DNS 10.16.136.1
ADD EBIPA INTERCONNECT DNS 10.16.255.2
ADD EBIPA INTERCONNECT DNS 10.16.255.3
SET EBIPA INTERCONNECT NTP PRIMARY 66.187.233.4
SET EBIPA INTERCONNECT NTP SECONDARY 64.73.32.134
ENABLE EBIPA INTERCONNECT 1,2,3,4
DISABLE EBIPA INTERCONNECT 5,6,7,8

#Uncomment following line to remove all user accounts currently in the system
#REMOVE USERS ALL

#Create Users
ADD USER "mlamouri"
SET USER CONTACT "mlamouri" ""
SET USER FULLNAME "mlamouri" ""
SET USER ACCESS "mlamouri" USER
ASSIGN INTERCONNECT 1,2,3,4,5,6,7,8 "mlamouri"
ASSIGN OA "mlamouri"
ENABLE USER "mlamouri"
ADD USER "spr"
SET USER CONTACT "spr" ""
SET USER FULLNAME "spr" ""
SET USER ACCESS "spr" ADMINISTRATOR
ASSIGN INTERCONNECT 1,2,3,4,5,6,7,8 "spr"
ASSIGN OA "spr"
ENABLE USER "spr"
ADD USER "testuser"
SET USER CONTACT "testuser" ""
SET USER FULLNAME "testuser" ""
SET USER ACCESS "testuser" OPERATOR
ENABLE USER "testuser"
ADD USER "tim"
SET USER CONTACT "tim" ""
SET USER FULLNAME "tim" "Tim Wilkinson"
SET USER ACCESS "tim" ADMINISTRATOR
ASSIGN INTERCONNECT 1,2,3,4,5,6,7,8 "tim"
ASSIGN OA "tim"
ENABLE USER "tim"

#Password Settings
DISABLE STRONG PASSWORDS
SET MINIMUM PASSWORD LENGTH 3

#Session Timeout Settings
SET SESSION TIMEOUT 1440

#Set LDAP Information
SET LDAP SERVER ""
SET LDAP PORT 0
SET LDAP NAME MAP OFF
SET LDAP SEARCH 1 ""
SET LDAP SEARCH 2 ""
SET LDAP SEARCH 3 ""
SET LDAP SEARCH 4 ""
SET LDAP SEARCH 5 ""
SET LDAP SEARCH 6 ""

#Uncomment following line to remove all LDAP accounts currently in the system
#REMOVE LDAP GROUP ALL
DISABLE LDAP

#Set SSO TRUST MODE
SET SSO TRUST Disabled

#Set Network Information
#NOTE: Setting your network information through a script while
#       remotely accessing the server could drop your connection.
#     If your connection is dropped this script may not execute to conclusion.
# SET OA NAME 2 ra-c7000-01-oa2
SET IPCONFIG STATIC 2 10.16.136.254 255.255.248.0 10.16.143.254 0.0.0.0 0.0.0.0
SET NIC AUTO 2
# SET OA NAME 1 ra-c7000-01-oa1
SET IPCONFIG STATIC 1 10.16.136.255 255.255.248.0 10.16.143.254 0.0.0.0 0.0.0.0
SET NIC AUTO 1
```

The Virtual Connection configuration as reported using *show*:

```
-> show all
****************************************************************************
 DEVICEBAY INFORMATION
****************************************************************************
============================================================================
ID          Enclosure        Bay   Device                 Profile
============================================================================
enc0:1      ra-c7000-01      1     ProLiant BL460c G6     mgmt_node1
enc0:2      ra-c7000-01      2     ProLiant BL460c G6     mgmt_node2
enc0:3      ra-c7000-01      3     ProLiant BL460c G6     host1
enc0:4      ra-c7000-01      4     ProLiant BL460c G6     <Unassigned>
enc0:5      ra-c7000-01      5     ProLiant BL460c G6     <Unassigned>
enc0:6      ra-c7000-01      6     ProLiant BL460c G6     <Unassigned>
enc0:7      ra-c7000-01      7     ProLiant BL460c G6     <Unassigned>
enc0:8      ra-c7000-01      8     ProLiant BL460c G6     <Unassigned>
enc0:9      ra-c7000-01      9     ProLiant BL460c G6     <Unassigned>
enc0:10     ra-c7000-01      10    ProLiant BL460c G6     test1
enc0:11     ra-c7000-01      11    ProLiant BL460c G6     <Unassigned>
enc0:12     ra-c7000-01      12    ProLiant BL460c G6     <Unassigned>
enc0:13     ra-c7000-01      13    ProLiant BL460c G6     <Unassigned>
enc0:14     ra-c7000-01      14    ProLiant BL460c G6     <Unassigned>
enc0:15     ra-c7000-01      15    ProLiant BL460c G6     <Unassigned>
enc0:16     ra-c7000-01      16    ProLiant BL460c G6     standalone

****************************************************************************
 DOMAIN INFORMATION
****************************************************************************

Domain Name       : ra-c7000-01_vc_domain
Checkpoint Status : Valid

----------------------------------------------------------------------------
Virtual Connect Domain IP Address Settings
----------------------------------------------------------------------------
Domain IP Status : Enabled
IP Address       : 10.16.136.229
Subnet Mask      : 255.255.248.0
Gateway          : 16.16.143.254

----------------------------------------------------------------------------
Ethernet MAC Address Settings
```

```
----------------------------------------------------------------------------
MAC Address Type : VC-Defined
Pool ID          : 10
Address Start    : 00-17-A4-77-24-00
Address End      : 00-17-A4-77-27-FF

----------------------------------------------------------------------------
Fibre Channel WWN Address Settings
----------------------------------------------------------------------------
WWN Address Type : VC-Defined
Pool ID          : 10
Address Start    : 50:06:0B:00:00:C2:86:00
Address End      : 50:06:0B:00:00:C2:89:FF


****************************************************************************
 ENCLOSURE INFORMATION
****************************************************************************
========================================================================
ID    Name         Import Status  Serial Number  Part         Asset Tag
                                                 Number
========================================================================
enc0  ra-c7000-01  Imported       USE010Y7JE     507019-B21

****************************************************************************
 ENET-CONNECTION INFORMATION
****************************************************************************
=========================================================================
Profile     Port  Network          PXE      MAC Address      Allocated  Status
                  Name                                       Speed
=========================================================================
host1       1     public           UseBIOS  00-17-A4-77-24-08  10Gb        OK
-------------------------------------------------------------------------
host1       2     <Unassigned>     UseBIOS  00-17-A4-77-24-0A  -- --       OK
-------------------------------------------------------------------------
mgmt_node1  1     public           UseBIOS  00-17-A4-77-24-00  10Gb        OK
-------------------------------------------------------------------------
mgmt_node1  2     mgmt-clus-inter  UseBIOS  00-17-A4-77-24-02  2.5Gb       OK
-------------------------------------------------------------------------
mgmt_node2  1     public           UseBIOS  00-17-A4-77-24-04  10Gb        OK
-------------------------------------------------------------------------
mgmt_node2  2     mgmt-clus-inter  UseBIOS  00-17-A4-77-24-06  2.5Gb       OK
-------------------------------------------------------------------------
standalone  1     public           UseBIOS  00-17-A4-77-24-0C  10Gb        OK
-------------------------------------------------------------------------
standalone  2     public           UseBIOS  00-17-A4-77-24-0E  10Gb        OK
-------------------------------------------------------------------------
test1       1     public           UseBIOS  00-17-A4-77-24-10  10Gb        OK
-------------------------------------------------------------------------
test1       2     mgmt-clus-inter  UseBIOS  00-17-A4-77-24-12  2.5Gb       OK
-------------------------------------------------------------------------
```

```
****************************************************************************
 ENET-VLAN INFORMATION
****************************************************************************
VLAN Tag Control      : Tunnel
Shared Server VLAN ID : false
Preferred Speed       : Auto
Max Speed             : Unrestricted

***************************************************************************
 EXTERNAL-MANAGER INFORMATION
***************************************************************************
No external manager exists

***************************************************************************
 FABRIC INFORMATION
***************************************************************************
========================================================
Name                Bay  Ports  Status  Speed  Login
========================================================
ra-c7000-01-fcvc01  3    1      OK      Auto   Dynamic
ra-c7000-01-fcvc02  4    1      OK      Auto   Dynamic

***************************************************************************
 FC-CONNECTION INFORMATION
***************************************************************************
===========================================================================
Profile     Port  Fabric Name         Speed  WWPN                     Status
===========================================================================
host1       1     ra-c7000-01-fcvc01  Auto   50:06:0B:00:00:C2:86:08  OK
---------------------------------------------------------------------------
host1       2     ra-c7000-01-fcvc02  Auto   50:06:0B:00:00:C2:86:0A  OK
---------------------------------------------------------------------------
mgmt_node1  1     ra-c7000-01-fcvc01  Auto   50:06:0B:00:00:C2:86:00  OK
---------------------------------------------------------------------------
mgmt_node1  2     ra-c7000-01-fcvc02  Auto   50:06:0B:00:00:C2:86:02  OK
---------------------------------------------------------------------------
mgmt_node2  1     ra-c7000-01-fcvc01  Auto   50:06:0B:00:00:C2:86:04  OK
---------------------------------------------------------------------------
mgmt_node2  2     ra-c7000-01-fcvc02  Auto   50:06:0B:00:00:C2:86:06  OK
---------------------------------------------------------------------------
standalone  1     <Unassigned>        Auto   50:06:0B:00:00:C2:86:0C  OK
---------------------------------------------------------------------------
standalone  2     <Unassigned>        Auto   50:06:0B:00:00:C2:86:0E  OK
---------------------------------------------------------------------------
test1       1     ra-c7000-01-fcvc01  Auto   50:06:0B:00:00:C2:86:10  OK
---------------------------------------------------------------------------
test1       2     ra-c7000-01-fcvc02  Auto   50:06:0B:00:00:C2:86:12  OK
---------------------------------------------------------------------------

****************************************************************************
 FIRMWARE INFORMATION
****************************************************************************
```

```
======================================================================
ID      Enclosure    Bay  Type     Firmware Version           Status
======================================================================
enc0:1  ra-c7000-01  1    VC-ENET  2.32 2010-01-06T02:07:47Z  OK
enc0:2  ra-c7000-01  2    VC-ENET  2.32 2010-01-06T02:07:47Z  OK
enc0:3  ra-c7000-01  3    VC-FC    1.01 v6.1.0_36             OK
enc0:4  ra-c7000-01  4    VC-FC    1.01 v6.1.0_36             OK

****************************************************************************
 IGMP INFORMATION
****************************************************************************
Enabled : false
Timeout : 260

****************************************************************************
 INTERCONNECT INFORMATION
****************************************************************************
========================================================================
ID      Enclosure    Bay  Type     Product Name                 Status
========================================================================
enc0:1  ra-c7000-01  1    VC-ENET  HP VC Flex-10 Enet Module    OK
enc0:2  ra-c7000-01  2    VC-ENET  HP VC Flex-10 Enet Module    OK
enc0:3  ra-c7000-01  3    VC-FC    HP VC 8Gb 24-Port FC Module  OK
enc0:4  ra-c7000-01  4    VC-FC    HP VC 8Gb 24-Port FC Module  OK

****************************************************************************
 LDAP INFORMATION
****************************************************************************
Enabled            : false
Local Users        : Enabled
NT Account Mapping : Disabled
Server Address     : -- --
SSL Port           : 636
Search Context 1   : -- --
Search Context 2   : -- --
Search Context 3   : -- --

****************************************************************************
 LDAP-CERTIFICATE INFORMATION
****************************************************************************
No LDAP certificates have been configured for the VC Domain

****************************************************************************
 LDAP-GROUP INFORMATION
****************************************************************************
No directory groups exist

****************************************************************************
 LOG-TARGET INFORMATION
****************************************************************************
No remote log destinations have been configured
```

```
****************************************************************************
 MAC-CACHE INFORMATION
****************************************************************************
Enabled          : true
Refresh Interval : 5

****************************************************************************
 NETWORK INFORMATION
****************************************************************************
===========================================================================
Name        Status  Shared      VLAN ID  Native      Private   Preferred
                    Uplink Set           VLAN                  Speed
===========================================================================
public      OK      -- --       -- --    Disabled    Disabled  Auto
---------------------------------------------------------------------------
mgmt-clus-inter  OK      -- --       -- --    Disabled    Disabled  2.5 GB
---------------------------------------------------------------------------

****************************************************************************
 PORT-MONITOR INFORMATION
****************************************************************************
----------------------------------------------------------------------------
Port Monitor Configuration
----------------------------------------------------------------------------
Enabled : false

****************************************************************************
 PROFILE INFORMATION
****************************************************************************
====================================================
Name        Device Bay  Server              Status
====================================================
mgmt_node1  enc0:1      ProLiant BL460c G6  OK
mgmt_node2  enc0:2      ProLiant BL460c G6  OK
host1       enc0:3      ProLiant BL460c G6  OK
standalone  enc0:16     ProLiant BL460c G6  OK
test1       enc0:10     ProLiant BL460c G6  OK
 Show call complete

****************************************************************************
 SERVER INFORMATION
****************************************************************************
=========================================================================
ID       Enclosure    Bay  Description      Status  Power  Profile
=========================================================================
enc0:1   ra-c7000-01  1    ProLiant         OK      On     mgmt_node1
                           BL460c G6
-------------------------------------------------------------------------
enc0:2   ra-c7000-01  2    ProLiant         OK      On     mgmt_node2
                           BL460c G6
-------------------------------------------------------------------------
enc0:3   ra-c7000-01  3    ProLiant         OK      On     host1
```

```
                           BL460c G6
-------------------------------------------------------------------------
enc0:4   ra-c7000-01  4    ProLiant         OK      Off    <Unassigned>
                           BL460c G6
-------------------------------------------------------------------------
enc0:5   ra-c7000-01  5    ProLiant         OK      Off    <Unassigned>
                           BL460c G6
-------------------------------------------------------------------------
enc0:6   ra-c7000-01  6    ProLiant         OK      Off    <Unassigned>
                           BL460c G6
-------------------------------------------------------------------------
enc0:7   ra-c7000-01  7    ProLiant         OK      Off    <Unassigned>
                           BL460c G6
-------------------------------------------------------------------------
enc0:8   ra-c7000-01  8    ProLiant         OK      Off    <Unassigned>
                           BL460c G6
-------------------------------------------------------------------------
enc0:9   ra-c7000-01  9    ProLiant         OK      Off    <Unassigned>
                           BL460c G6
-------------------------------------------------------------------------
enc0:10  ra-c7000-01  10   ProLiant         OK      Off     <Unassigned>
                           BL460c G6
-------------------------------------------------------------------------
enc0:11  ra-c7000-01  11   ProLiant         OK      Off    <Unassigned>
                           BL460c G6
-------------------------------------------------------------------------
enc0:12  ra-c7000-01  12   ProLiant         OK      Off    <Unassigned>
                           BL460c G6
-------------------------------------------------------------------------
enc0:13  ra-c7000-01  13   ProLiant         OK      Off    <Unassigned>
                           BL460c G6
-------------------------------------------------------------------------
enc0:14  ra-c7000-01  14   ProLiant         OK      Off    <Unassigned>
                           BL460c G6
-------------------------------------------------------------------------
enc0:15  ra-c7000-01  15   ProLiant         OK      Off    <Unassigned>
                           BL460c G6
-------------------------------------------------------------------------
enc0:16  ra-c7000-01  16   ProLiant         OK      Off    <Unassigned>
                           BL460c G6
-------------------------------------------------------------------------

****************************************************************************
 SERVERID INFORMATION
****************************************************************************
Serial Number Type : VC-Defined
Pool ID            : 10
Start              : VCX0000900
End                : VCX00009ZZ

****************************************************************************
 SERVER-PORT INFORMATION
```

```
****************************************************************************
==========================================================================
Port      Server   I/O      Adapter Type          ID          Profile
                   Module
==========================================================================
1 (Flex   enc0:1   1        -- --                 enc0:1:d1   -- --
NIC)
--------------------------------------------------------------------------
1 (Flex   enc0:10  1        -- --                 enc0:1:d10  -- --
NIC)
--------------------------------------------------------------------------
1         enc0:11  1        Flex-10 Embedded      enc0:1:d11  -- --
                            Ethernet
--------------------------------------------------------------------------
1         enc0:12  1        Flex-10 Embedded      enc0:1:d12  -- --
                            Ethernet
--------------------------------------------------------------------------
1         enc0:13  1        Flex-10 Embedded      enc0:1:d13  -- --
                            Ethernet
--------------------------------------------------------------------------
1         enc0:14  1        Flex-10 Embedded      enc0:1:d14  -- --
                            Ethernet
--------------------------------------------------------------------------
1         enc0:15  1        Flex-10 Embedded      enc0:1:d15  -- --
                            Ethernet
--------------------------------------------------------------------------
1 (Flex   enc0:16  1        -- --                 enc0:1:d16  -- --
NIC)
--------------------------------------------------------------------------
1 (Flex   enc0:2   1        -- --                 enc0:1:d2   -- --
NIC)
--------------------------------------------------------------------------
1 (Flex   enc0:3   1        -- --                 enc0:1:d3   -- --
NIC)
--------------------------------------------------------------------------
1         enc0:4   1        Flex-10 Embedded      enc0:1:d4   -- --
                            Ethernet
--------------------------------------------------------------------------
1         enc0:5   1        Flex-10 Embedded      enc0:1:d5   -- --
                            Ethernet
--------------------------------------------------------------------------
1         enc0:6   1        Flex-10 Embedded      enc0:1:d6   -- --
                            Ethernet
--------------------------------------------------------------------------
1         enc0:7   1        Flex-10 Embedded      enc0:1:d7   -- --
                            Ethernet
--------------------------------------------------------------------------
1         enc0:8   1        Flex-10 Embedded      enc0:1:d8   -- --
                            Ethernet
--------------------------------------------------------------------------
1         enc0:9   1        Flex-10 Embedded      enc0:1:d9   -- --
                            Ethernet
```

```
--------------------------------------------------------------------------
2 (Flex   enc0:1   2        -- --                 enc0:2:d1   -- --
NIC)
--------------------------------------------------------------------------
2 (Flex   enc0:10  2        -- --                 enc0:2:d10  -- --
NIC)
--------------------------------------------------------------------------
2         enc0:11  2        Flex-10 Embedded      enc0:2:d11  -- --
                            Ethernet
--------------------------------------------------------------------------
2         enc0:12  2        Flex-10 Embedded      enc0:2:d12  -- --
                            Ethernet
--------------------------------------------------------------------------
2         enc0:13  2        Flex-10 Embedded      enc0:2:d13  -- --
                            Ethernet
--------------------------------------------------------------------------
2         enc0:14  2        Flex-10 Embedded      enc0:2:d14  -- --
                            Ethernet
--------------------------------------------------------------------------
2         enc0:15  2        Flex-10 Embedded      enc0:2:d15  -- --
                            Ethernet
--------------------------------------------------------------------------
2 (Flex   enc0:16  2        -- --                 enc0:2:d16  -- --
NIC)
--------------------------------------------------------------------------
2 (Flex   enc0:2   2        -- --                 enc0:2:d2   -- --
NIC)
--------------------------------------------------------------------------
2 (Flex   enc0:3   2        -- --                 enc0:2:d3   -- --
NIC)
--------------------------------------------------------------------------
2         enc0:4   2        Flex-10 Embedded      enc0:2:d4   -- --
                            Ethernet
--------------------------------------------------------------------------
2         enc0:5   2        Flex-10 Embedded      enc0:2:d5   -- --
                            Ethernet
--------------------------------------------------------------------------
2         enc0:6   2        Flex-10 Embedded      enc0:2:d6   -- --
                            Ethernet
--------------------------------------------------------------------------
2         enc0:7   2        Flex-10 Embedded      enc0:2:d7   -- --
                            Ethernet
--------------------------------------------------------------------------
2         enc0:8   2        Flex-10 Embedded      enc0:2:d8   -- --
                            Ethernet
--------------------------------------------------------------------------
2         enc0:9   2        Flex-10 Embedded      enc0:2:d9   -- --
                            Ethernet
--------------------------------------------------------------------------
1         enc0:1   3        QLogic QMH2562 8Gb    enc0:3:d1   mgmt_node1
                            FC HBA for HP
                            BladeSystem c-Class
```

```
-------------------------------------------------------------------------
1         enc0:10  3        QLogic QMH2562 8Gb    enc0:3:d10  test1
                            FC HBA for HP
                            BladeSystem c-Class
-------------------------------------------------------------------------
1         enc0:11  3        QLogic QMH2562 8Gb    enc0:3:d11  -- --
                            FC HBA for HP
                            BladeSystem c-Class
-------------------------------------------------------------------------
1         enc0:12  3        QLogic QMH2562 8Gb    enc0:3:d12  -- --
                            FC HBA for HP
                            BladeSystem c-Class
-------------------------------------------------------------------------
1         enc0:13  3        QLogic QMH2562 8Gb    enc0:3:d13  -- --
                            FC HBA for HP
                            BladeSystem c-Class
-------------------------------------------------------------------------
1         enc0:14  3        QLogic QMH2562 8Gb    enc0:3:d14  -- --
                            FC HBA for HP
                            BladeSystem c-Class
-------------------------------------------------------------------------
1         enc0:15  3        QLogic QMH2562 8Gb    enc0:3:d15  -- --
                            FC HBA for HP
                            BladeSystem c-Class
-------------------------------------------------------------------------
1         enc0:16  3        QLogic QMH2562 8Gb    enc0:3:d16  -- --
                            FC HBA for HP
                            BladeSystem c-Class
-------------------------------------------------------------------------
1         enc0:2   3        QLogic QMH2562 8Gb    enc0:3:d2   mgmt_node2
                            FC HBA for HP
                            BladeSystem c-Class
-------------------------------------------------------------------------
1         enc0:3   3        QLogic QMH2562 8Gb    enc0:3:d3   host1
                            FC HBA for HP
                            BladeSystem c-Class
-------------------------------------------------------------------------
1         enc0:4   3        QLogic QMH2562 8Gb    enc0:3:d4   -- --
                            FC HBA for HP
                            BladeSystem c-Class
-------------------------------------------------------------------------
1         enc0:5   3        QLogic QMH2562 8Gb    enc0:3:d5   -- --
                            FC HBA for HP
                            BladeSystem c-Class
-------------------------------------------------------------------------
1         enc0:6   3        QLogic QMH2562 8Gb    enc0:3:d6   -- --
                            FC HBA for HP
                            BladeSystem c-Class
-------------------------------------------------------------------------
1         enc0:7   3        QLogic QMH2562 8Gb    enc0:3:d7   -- --
                            FC HBA for HP
                            BladeSystem c-Class
```

```
--------------------------------------------------------------------------
1         enc0:8   3        QLogic QMH2562 8Gb    enc0:3:d8   -- --
                            FC HBA for HP
                            BladeSystem c-Class
--------------------------------------------------------------------------
1         enc0:9   3        QLogic QMH2562 8Gb    enc0:3:d9   -- --
                            FC HBA for HP
                            BladeSystem c-Class
--------------------------------------------------------------------------
2         enc0:1   4        QLogic QMH2562 8Gb    enc0:4:d1   mgmt_node1
                            FC HBA for HP
                            BladeSystem c-Class
--------------------------------------------------------------------------
2         enc0:10  4        QLogic QMH2562 8Gb    enc0:4:d10  test1
                            FC HBA for HP
                            BladeSystem c-Class
--------------------------------------------------------------------------
2         enc0:11  4        QLogic QMH2562 8Gb    enc0:4:d11  -- --
                            FC HBA for HP
                            BladeSystem c-Class
--------------------------------------------------------------------------
2         enc0:12  4        QLogic QMH2562 8Gb    enc0:4:d12  -- --
                            FC HBA for HP
                            BladeSystem c-Class
--------------------------------------------------------------------------
2         enc0:13  4        QLogic QMH2562 8Gb    enc0:4:d13  -- --
                            FC HBA for HP
                            BladeSystem c-Class
--------------------------------------------------------------------------
2         enc0:14  4        QLogic QMH2562 8Gb    enc0:4:d14  -- --
                            FC HBA for HP
                            BladeSystem c-Class
--------------------------------------------------------------------------
2         enc0:15  4        QLogic QMH2562 8Gb    enc0:4:d15  -- --
                            FC HBA for HP
                            BladeSystem c-Class
--------------------------------------------------------------------------
2         enc0:16  4        QLogic QMH2562 8Gb    enc0:4:d16  -- --
                            FC HBA for HP
                            BladeSystem c-Class
--------------------------------------------------------------------------
2         enc0:2   4        QLogic QMH2562 8Gb    enc0:4:d2   mgmt_node2
                            FC HBA for HP
                            BladeSystem c-Class
--------------------------------------------------------------------------
2         enc0:3   4        QLogic QMH2562 8Gb    enc0:4:d3   host1
                            FC HBA for HP
                            BladeSystem c-Class
--------------------------------------------------------------------------
2         enc0:4   4        QLogic QMH2562 8Gb    enc0:4:d4   -- --
                            FC HBA for HP
                            BladeSystem c-Class
```

```
--------------------------------------------------------------------------
2         enc0:5   4        QLogic QMH2562 8Gb    enc0:4:d5   -- --
                            FC HBA for HP
                            BladeSystem c-Class
--------------------------------------------------------------------------
2         enc0:6   4        QLogic QMH2562 8Gb    enc0:4:d6   -- --
                            FC HBA for HP
                            BladeSystem c-Class
--------------------------------------------------------------------------
2         enc0:7   4        QLogic QMH2562 8Gb    enc0:4:d7   -- --
                            FC HBA for HP
                            BladeSystem c-Class
--------------------------------------------------------------------------
2         enc0:8   4        QLogic QMH2562 8Gb    enc0:4:d8   -- --
                            FC HBA for HP
                            BladeSystem c-Class
--------------------------------------------------------------------------
2         enc0:9   4        QLogic QMH2562 8Gb    enc0:4:d9   -- --
                            FC HBA for HP
                            BladeSystem c-Class
--------------------------------------------------------------------------

****************************************************************************
 SERVER-PORT-MAP INFORMATION
****************************************************************************
ERROR: Operation not allowed. The domain is configured to use VLAN Tunneling

****************************************************************************
 SNMP INFORMATION
****************************************************************************
===========================================================
Type    Enabled  Community Name  System Contact  SMIS
                                                 Enabled
===========================================================
Domain  true     public          -- --           -- --
Enet    true     public          -- --           -- --
FC      true     public          -- --           false

****************************************************************************
 SNMP-TRAP INFORMATION
****************************************************************************
No SNMP traps exist

****************************************************************************
 SSH INFORMATION
****************************************************************************
No SSH key exists in the VC Domain

****************************************************************************
 SSL INFORMATION
****************************************************************************
Strength : All
```

```
****************************************************************************
 SSL-CERTIFICATE INFORMATION
****************************************************************************
============================================================================
Serial Number            Issuer                     Subject
============================================================================
80:39:6F:C6:A7:7B:A8:E3  VCEXTW29520140:Virtual     VCEXTW29520140:Virtual
                         Connect                    Connect
                         Manager:Hewlett-Packard    Manager:Hewlett-Packard
----------------------------------------------------------------------------

****************************************************************************
 STACKINGLINK INFORMATION
****************************************************************************
Connection Status : OK
Redundancy Status : OK

===========================================
Link  Speed  Connected From  Connected To
===========================================
1     10Gb   enc0:1:X8       enc0:2:X8

****************************************************************************
 STATUS INFORMATION
****************************************************************************
Overall Domain Status : Normal : The domain is fully functional.

Critical    : 0
Major       : 0
Minor       : 0
Warning     : 0
Information : 0
Unknown     : 0


****************************************************************************
 UPLINKPORT INFORMATION
****************************************************************************
==========================================================================
ID         Enclosure    Status           Type     Speed  Used By
==========================================================================
enc0:1:X1  ra-c7000-01  Not Linked       CX4      Auto   -- --
--------------------------------------------------------------------------
enc0:1:X2  ra-c7000-01  Not Linked       absent   Auto   -- --
--------------------------------------------------------------------------
enc0:1:X3  ra-c7000-01  Not Linked       absent   Auto   -- --
--------------------------------------------------------------------------
enc0:1:X4  ra-c7000-01  Not Linked       absent   Auto   -- --
--------------------------------------------------------------------------
enc0:1:X5  ra-c7000-01  Not Linked       absent   Auto   -- --
--------------------------------------------------------------------------
```

```
enc0:1:X6  ra-c7000-01  Not Linked       absent   Auto   -- --
---------------------------------------------------------------------------
enc0:1:X7  ra-c7000-01  Linked           SFP-DAC  Auto   public
                        (Active) (10Gb)
---------------------------------------------------------------------------
enc0:2:X1  ra-c7000-01  Not Linked       CX4      Auto   -- --
---------------------------------------------------------------------------
enc0:2:X2  ra-c7000-01  Not Linked       absent   Auto   -- --
---------------------------------------------------------------------------
enc0:2:X3  ra-c7000-01  Not Linked       absent   Auto   -- --
---------------------------------------------------------------------------
enc0:2:X4  ra-c7000-01  Not Linked       absent   Auto   -- --
---------------------------------------------------------------------------
enc0:2:X5  ra-c7000-01  Not Linked       absent   Auto   -- --
---------------------------------------------------------------------------
enc0:2:X6  ra-c7000-01  Not Linked       absent   Auto   -- --
---------------------------------------------------------------------------
enc0:2:X7  ra-c7000-01  Linked (10Gb)    SFP-DAC  Auto   -- --
---------------------------------------------------------------------------
enc0:3:1   ra-c7000-01  Logged In        -- --    Auto   ra-c7000-01-fcvc01
---------------------------------------------------------------------------
enc0:3:2   ra-c7000-01  Not Logged In    -- --    Auto   -- --
---------------------------------------------------------------------------
enc0:3:3   ra-c7000-01  Not Logged In    -- --    Auto   -- --
---------------------------------------------------------------------------
enc0:3:4   ra-c7000-01  Not Logged In    -- --    Auto   -- --
---------------------------------------------------------------------------
enc0:3:5   ra-c7000-01  Not Logged In    -- --    Auto   -- --
---------------------------------------------------------------------------
enc0:3:6   ra-c7000-01  Not Logged In    -- --    Auto   -- --
---------------------------------------------------------------------------
enc0:3:7   ra-c7000-01  Not Logged In    -- --    Auto   -- --
---------------------------------------------------------------------------
enc0:3:8   ra-c7000-01  Not Logged In    -- --    Auto   -- --
---------------------------------------------------------------------------
enc0:4:1   ra-c7000-01  Logged In        -- --    Auto   ra-c7000-01-fcvc02
---------------------------------------------------------------------------
enc0:4:2   ra-c7000-01  Not Logged In    -- --    Auto   -- --
---------------------------------------------------------------------------
enc0:4:3   ra-c7000-01  Not Logged In    -- --    Auto   -- --
---------------------------------------------------------------------------
enc0:4:4   ra-c7000-01  Not Logged In    -- --    Auto   -- --
---------------------------------------------------------------------------
enc0:4:5   ra-c7000-01  Not Logged In    -- --    Auto   -- --
---------------------------------------------------------------------------
enc0:4:6   ra-c7000-01  Not Logged In    -- --    Auto   -- --
---------------------------------------------------------------------------
enc0:4:7   ra-c7000-01  Not Logged In    -- --    Auto   -- --
---------------------------------------------------------------------------
enc0:4:8   ra-c7000-01  Not Logged In    -- --    Auto   -- --
---------------------------------------------------------------------------
```

```
****************************************************************************
 UPLINKSET INFORMATION
****************************************************************************
No shared uplink port sets exist

****************************************************************************
 USER INFORMATION
****************************************************************************
=================================================================
User Name      Privileges  Full Name      Contact Info  Enabled
=================================================================
Administrator  domain      -- --          -- --         true
               server
               network
               storage
-----------------------------------------------------------------
spr            domain      -- --          -- --         true
               server
               network
               storage
-----------------------------------------------------------------
mark           domain      -- --          -- --         true
               server
               network
               storage
-----------------------------------------------------------------
tim            domain      -- --          -- --         true
               server
               network
               storage
-----------------------------------------------------------------

****************************************************************************
 USER-SECURITY INFORMATION
****************************************************************************
Strong Passwords        : Disabled
Minimum Password Length : 3

****************************************************************************
 VERSION INFORMATION
****************************************************************************
HP Virtual Connect Management CLI v2.32
(C) Copyright 2006-2009 Hewlett-Packard Development Company, L.P.
All Rights Reserved
```

A.7 Java

The javaApp RPM is the RPM that was created and detailed in the **Red Hat Cloud Foundations Edition One: Private IaaS Clouds** document.

Appendix B: Bugzillas

The following Red Hat bugzilla reports were open(ed) issues at the time of this exercise.

1. BZ 518531 - Need "disable_tpa" parameter with bnx2x to get networking to work for guests
   https://bugzilla.redhat.com/show_bug.cgi?id=518531
2. BZ 593048 - Intermittent "could not query memory balloon allocation" errors with virt-install
   https://bugzilla.redhat.com/show_bug.cgi?id=593048
3. BZ 593093 - not all cobbler kickstart variables correctly applied
   https://bugzilla.redhat.com/show_bug.cgi?id=593093
4. BZ 602402 - bnx2x panic dumps with multiple interfaces enabled
   https://bugzilla.redhat.com/show_bug.cgi?id=602402
5. BZ 617725 - Using add-vm powershell command fails to take templates display type
   https://bugzilla.redhat.com/show_bug.cgi?id=617725
6. BZ 617730 - Can not use -DisplayType to set to VNC on add-vm
   https://bugzilla.redhat.com/show_bug.cgi?id=617730